EXCEL AVANÇADO

Gilberto Cardoso Parreira

Índice analítico

Introdução

O presente trabalho procura apresentar, de uma maneira o mais prática possível, um aplicativo que é presença constante em quase todos os computadores hoje em dia mas que, apesar disso, permanece com grande parte de suas reais capacidades ignoradas: o Microsoft Excel.

O fato é que muitas vezes gastamos um tempo precioso em tarefas repetitivas de manipulação de dados e cálculos que poderiam ser realizados em uma fração apreciável deste tempo, desde que nos debrucemos sobre algumas ferramentas de uso até simples, mas que permanecem escondidas atrás de opções de menus que não utilizamos freqüentemente ou até mesmo do medo de enfrentar algumas linhas de programação, como se fosse isto tarefa para iniciados deste mundo de linguagens e códigos.

Durante o estudo deste trabalho, buscaremos desmistificar as ferramentas mais eficazes para introdução de dados em tabelas, busca, filtragem e resumo destes dados, visando extrair a informação oculta por trás de um emaranhado de valores e identificadores.

Como será visto, são ferramentas simples, mas não tão simples que não requeiram uma experiência anterior na utilização básica do Microsoft Excel, em seus conceitos de utilização de planilhas, formatação de dados, criação de fórmulas para cálculos e geração de gráficos. Todos estes assuntos serão considerados como alicerce inicial para este estudo.

"You never finish a spreadsheet. You just stop working on it."

Utilizando tabelas

No Microsoft Excel, pode-se resumir ou analisar valores em uma lista de várias maneiras:

- Para calcular o valor de total geral de uma coluna ou linha de dados, pode-se usar o recurso *AutoSoma* para criar rapidamente uma fórmula que some os valores na coluna ou linha.
- Pode-se calcular automaticamente *subtotais* de valores em uma lista. Por exemplo, se você tiver uma lista que contém valores de vendas de vendedores de diferentes regiões, pode-se calcular valores de subtotal para cada vendedor ou cada região.
- Para calcular o valor total de linhas em uma lista que atendam a uma condição específica, pode-se usar o comando *AutoFiltro* para exibir as linhas que atendem à condição e usar, em seguida, o recurso *AutoSoma* para calcular o total para as linhas visíveis apenas.
- Para resumir somente os valores que atendem a critérios complexos, como "fornecer uma conta das linhas ou registros em que as vendas foram superior a 1.000 mas inferiores a 2.500", use uma *função de banco de dados*. Para usar uma função de banco de dados, você deve criar um intervalo de critérios para especificar a condição a ser atendida.
- Pode-se criar uma fórmula que calcule totais para os valores de uma lista, com base em uma condição específica, usando o *Assistente de soma condicional*.
- Pode-se resumir os valores de uma lista de um relatório que use métodos de cálculo e formatações especificadas por você através de uma *Tabela dinâmica*.

Diretrizes para criar uma lista em uma planilha

O Microsoft Excel oferece uma série de recursos que facilitam o gerenciamento e a análise de dados em uma lista. Para utilizar estes recursos, insira dados em uma lista de acordo com as seguintes diretrizes.

Tamanho e localização da lista

- Evite ter mais de uma lista em cada planilha, pois alguns recursos de gerenciamento de lista, como filtro, só podem ser usados em uma lista de cada vez.
- Deixe pelo menos uma coluna em branco e uma linha em branco entre a lista e os outros dados da planilha. Este procedimento ajuda o Microsoft Excel a selecionar a lista quando você classificar, filtrar ou inserir subtotais automáticos.
- Evite colocar linhas e colunas em branco na lista para que o Microsoft Excel detecte e selecione a lista com mais facilidade.
- Evite colocar dados essenciais à esquerda ou à direita da lista; os dados poderão ficar ocultos quando você filtrar a lista.

Rótulos de coluna

- Crie rótulos de coluna na primeira linha da lista. O Microsoft Excel usa estes rótulos para criar relatórios e para localizar e organizar dados.
- Use estilos de fonte, alinhamento, formato, padrão, borda ou maiúsculas para os rótulos de coluna que sejam diferentes do formato atribuído aos dados da lista.
- Use bordas de célula, e não linhas em branco ou tracejadas, para inserir linhas abaixo dos rótulos quando desejar separar os rótulos dos dados.

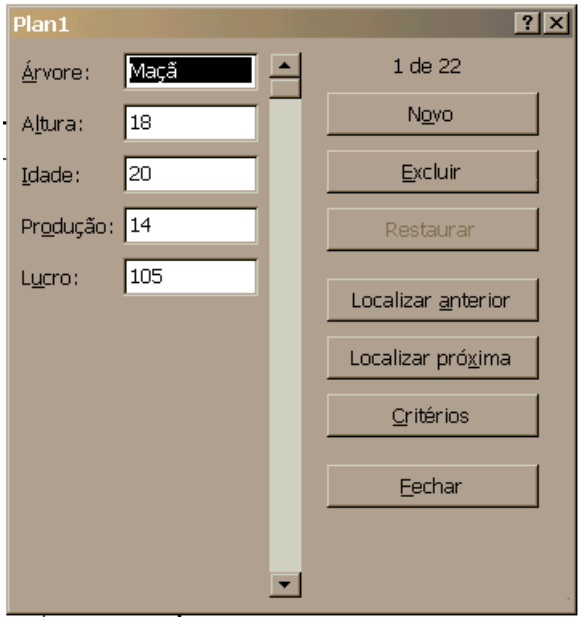
Conteúdo das linhas e colunas

- Elabore a sua lista de modo que todas as linhas contenham itens semelhantes na mesma coluna.
- Não insira espaços extras no início de uma célula; os espaços extras afetam os processos de classificação e localização.
- Não use uma linha em branco para separar rótulos de colunas da primeira linha de dados.

Introduzindo dados através de formulário

Uma das ferramentas mais simples e talvez menos utilizadas para tratamento de dados do Microsoft Excel é o formulário, que é utilizada normalmente para entrada de dados em uma planilha, mas que também pode ser utilizado em buscas simples.

Para se utilizar o formulário como entrada de dados basta selecionar os campos de uma lista onde se deseja introduzir dados e selecionar o comando *Formulário*, presente no menu *Dados*. Caso



não exista ainda nenhum dado inserido, basta digitar os rótulos de identificação das colunas onde serão introduzidos dados e acionar o comando *Formulário* para começar uma nova lista.

Será exibido então um formulário, como mostrado acima, contendo todos os campos selecionados da tabela. Clicando-se em *Novo*, é criado um novo registro limpo que pode ser utilizado para a introdução de dados. A partir daí, pode-se ir digitando os dados e teclando-se TAB após cada

introdução. Ao final dos dados, o cursor voltará para a tecla *Novo*. Caso se queira continuar a introduzir dados, basta teclar ENTER neste momento e retornar a digitação.

Pode-se também eliminar linhas da tabela através deste formulário, utilizando-se a tecla *Excluir*. Cada exclusão movimentará os dados remanescentes nas colunas para cima em uma linha. Assim, deve-se ter muito cuidado ao eliminar células quando nem todos as colunas de uma lista estão selecionadas, já que a exclusão provocará a perda da associação entre os dados.

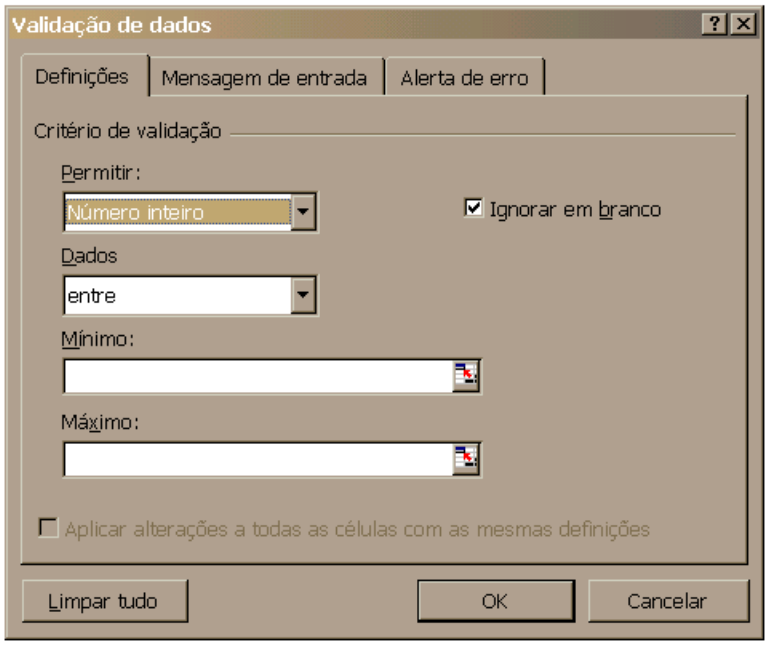
Uma característica menos conhecida desta ferramenta é sua capacidade de funcionar como ferramenta de filtro. Esta função pode ser acionada utilizando-se a tecla *Critérios*. Neste momento, será exibido um registro em branco onde o usuário poderá introduzir critérios de seleção de registros (exemplos: > 10, <=1000, <> "Carlos", etc.) em determinados campos. Note que é possível especificar mais de um critério ao mesmo tempo. Desta maneira, pode-se criar uma regra que procure aqueles registros que possuam Altura > 1,50 e Idade > 16, por exemplo. Após definir os critérios, basta utilizar a tecla *Formulário* para retornar ao formulário de entrada de dados. Através das teclas *Localizar anterior* e *Localizar próxima* pode-se acessar todos os registros que atendam aos critérios especificados.

Validando entrada de dados

Complementando o conjunto de ferramentas de entradas de dados, é possível também validar as entradas de dados no Microsoft Excel, ou seja, indicar quais são os valores possíveis para cada campo de uma lista. Existe a opção de se exibir mensagens de entrada de dados que orientem o usuário sobre os valores que foram especificados para o campo e até mesmo criar mensagens de erro que alertem sobre valores introduzidos erroneamente. Todos estes ajustes são feitos através do comando *Validação*, presente no menu *Dados*.

Restringir entradas de células a números, datas ou horas dentro de limites especificados

1. Selecione as células que você deseja restringir.
2. No menu Dados, clique em Validação e depois clique na guia Definições.
3. Na caixa *Permitir*, clique no tipo de dados.
   Para especificar somente números, clique em *Número inteiro ou Decimal*.

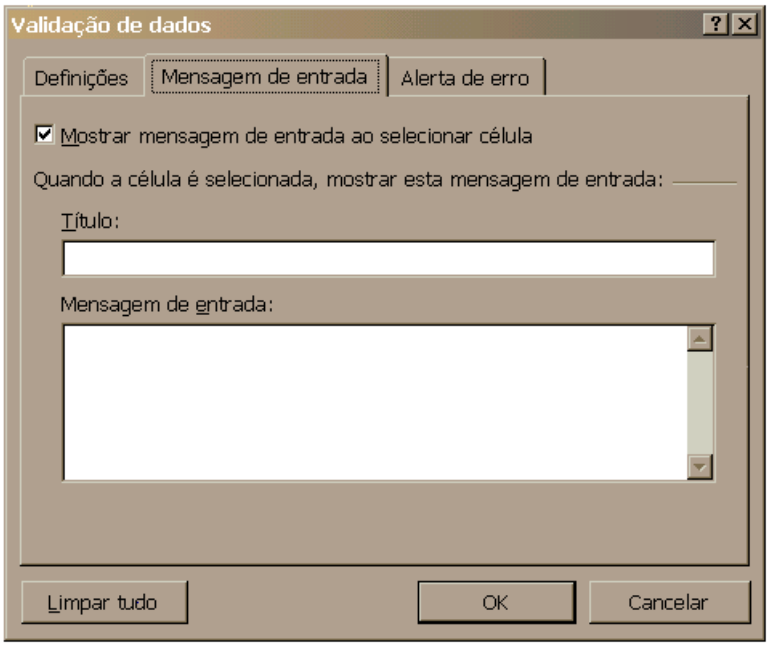


Para especificar somente datas ou horas, clique em *Data ou Hora*.

4. Clique no operador que você deseja na caixa *Dados* e especifique o limite superior ou inferior para os dados, ou ambos os limites, dependendo do operador selecionado. Pode-se inserir valores, referências de células ou fórmulas para os limites.

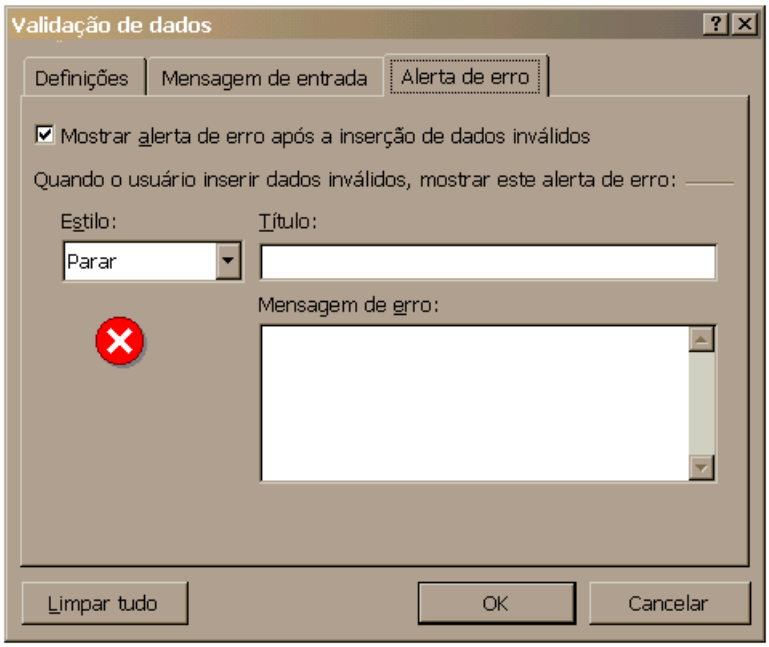
   Se você desejar permitir que a célula que você está restringindo fique em branco, ou se você definir limites que usam uma referência de célula ou fórmula que depende de células que estejam inicialmente em branco, certifique-se de que *Ignorar em branco* esteja marcada.

   Para impor as restrições que você definiu para as células em branco, tratando essas células



   como se contivessem zeros, desmarque a caixa de seleção *Ignorar em branco*.

5. Para exibir mensagens que solicitam entradas e para explicar ou evitar entradas incorretas,



especifique os tipos de mensagens que você deseja nas guias *Mensagem de entrada* e *Alerta de erro*.

6. Para desativar a exibição de mensagens, desmarque a caixa de seleção *Mostrar mensagem de entrada ao selecionar célula* na guia *Mensagem de entrada* e desmarque a caixa de seleção *Mostrar alerta de erro após ser inserido dado inválido* na guia *Alerta de erro*.

Observações

- Quando você especificar o tipo de dados permissível, a formatação da célula não será afetada. Para formatar as células como números, datas ou horas, clique em *Células* no menu *Formatar* e clique, em seguida, na guia *Número*. Selecione o formato desejado na caixa *Categoria* e depois selecione as opções desejadas.
- Uma fórmula para um limite pode avaliar dados apenas na planilha em que você configurou as restrições. Para usar os dados contidos em outras planilhas ou pastas de trabalho em uma fórmula, insira uma referência para os dados externos em uma célula na planilha ativa, ou defina um nome para os dados externos na planilha ativa. A fórmula poderá, então, referir-se à célula ou ao nome na mesma planilha. Por exemplo, se os dados que você deseja usar em uma fórmula estiverem na célula A6 na primeira planilha em uma pasta de trabalho denominada Despesas.xls, você poderá definir o nome *DadosVálidos* na planilha ativa como =[Despesas.xls]Plan1!$A$6 e inserir uma referência para *DadosVálidos* quando você especificar os limites para os dados.
- Você não poderá usar constantes matriciais em fórmulas de restrição de dados.

Tipos de restrições de dados

As opções exibidas na caixa de diálogo *Validação* de dados se alteram em função de suas seleções nas caixas *Permitir* e *Dados*.

Tipo	Descrição e opções
Qualquer valor	Não impõe restrições quanto às entradas válidas. Use esta definição quando você desejar exibir uma mensagem de entrada sem verificar as entradas válidas.
Personalizada	Permite que você insira uma fórmula, use uma expressão ou refira-se a um cálculo em outra célula para determinar entradas válidas. Fórmula - Insira uma fórmula, iniciando com um sinal de igual (=). A fórmula deverá retornar Verdadeiro ou Falso. Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas.
Data	Especifica que as entradas devem ser datas. Dados - Clique em um operador e preencha as opções. Data - Insira a data para o operador. Data inicial - Insira a primeira data para o operador. Data final - Insira a última data para o operador. Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas.
Decimal	Especifica que as entradas devem ser números ou frações. Dados - Clique em um operador e preencha as opções.

	Valor - Insira o número para o operador. Mínimo - Insira o menor número para o operador. Máximo - Insira o maior número para o operador. Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas.
Lista	Permite que você especifique uma lista das entradas válidas. Origem - Insira uma referência para o intervalo na pasta de trabalho onde você inseriu as entradas válidas, insira uma referência a um nome definido, ou digite as entradas válidas, separadas por vírgulas (por exemplo, inferior, médio, superior) Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas. Dropdown na célula - Marque para exibir uma seta drop-down que fornece a lista em que se deve selecionar, quando a pessoa inserindo os dados clicar na célula. Desmarque para omitir a lista drop-down.
Comprimento do texto	Especifica o número de caracteres para as entradas. Dados - Clique em um operador e preencha as opções. Comprimento - Insira o número para o operador. Mínimo - Insira o menor número para o operador. Máximo - Insira o maior número para o operador. Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas.
Hora	Especifica que as entradas devem ser horas. Dados - Clique em um operador e preencha as opções. Hora - Insira a hora para o operador. Hora inicial - Insira a hora mais antiga para o operador. Hora final - Insira a hora mais recente para o operador. Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas.
Número inteiro	Especifica que as entradas devem ser inteiros. Dados - Clique em um operador e preencha as opções. Valor - Insira o número para o operador. Mínimo - Insira o menor número para o operador. Máximo - Insira o maior número para o operador. Ignorar em branco - Marque para permitir entradas em branco na célula. Desmarque para especificar que as entradas em branco não são válidas.

Operadores de validação de dados

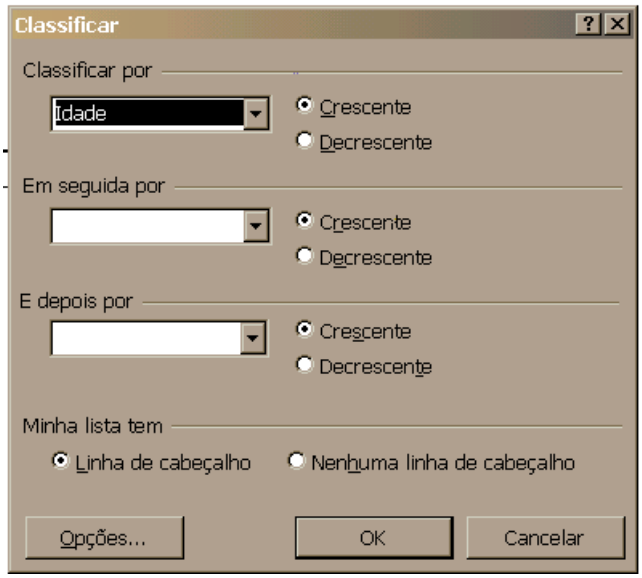
Nas caixas de opções, pode-se inserir valores, referências de células ou fórmulas.

Operador	Opções
entre	As entradas válidas são maiores ou iguais à entrada atribuída à opção Mínimo, Data inicial ou Hora inicial e menores ou iguais à entrada atribuída à opção Máximo, Data final ou Hora final.
não está entre	As entradas válidas são menores do que a entrada atribuída à opção Mínimo, Data inicial ou Hora inicial ou maiores do que a entrada atribuída à opção Máximo, Data final ou Hora final.
igual a	As entradas válidas coincidem com a entrada atribuída à opção Valor, Comprimento, Data ou Hora.
diferente de	As entradas válidas não coincidem com a entrada atribuída à opção Valor, Comprimento, Data ou Hora.
maior do que	As entradas válidas ultrapassam a entrada atribuída à opção Mínimo, Data inicial ou Hora inicial.
menor do que	As entradas válidas estão abaixo da entrada atribuída à opção Máximo, Data final ou Hora final.
maior ou igual a	As entradas válidas coincidem ou ultrapassam a entrada atribuída à opção Mínimo, Data inicial ou Hora inicial.
menor ou igual a	As entradas válidas coincidem ou ficam abaixo da entrada atribuída à opção Máximo, Data final ou Hora final.

Inserindo subtotais em uma lista

Muitas vezes desejamos obter segmentações de uma lista com os respectivos subtotais de cada segmento. Isto é bastante utilizado em vendas, por exemplo, quando necessitamos identificar o volume de vendas para cada região ou cidade. Uma aplicação industrial poderia ser descobrir o número de defeitos de cada categoria em uma amostragem da produção. Este tipo de tarefa é bastante comum e o Microsoft Excel possui uma ferramenta para automatiza-la com bastante facilidade. Para utilizá-la, siga os seguintes passos:

1. Primeiramente, classifique a lista pela coluna para a qual você deseja calcular subtotais. Por exemplo, para resumir as unidades vendidas por cada vendedor em uma lista de vendedores,

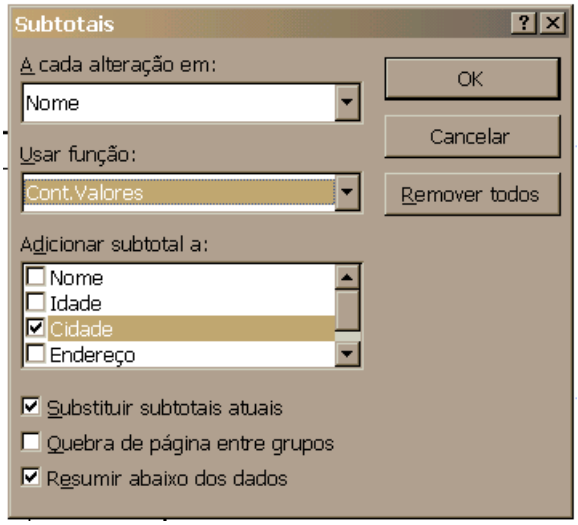


quantidades de venda e o número de unidades vendidas, classifique a lista pela coluna do vendedor.

Pode-se reordenar as linhas ou colunas de uma lista com base nos valores de uma lista através da classificação. Quando você classifica, o Microsoft Excel reordena as linhas, colunas ou células individuais usando a ordem de classificação especificada. Pode-se classificar listas em ordem crescente(1 a 9, A a Z) ou decrescente (9 a 1, Z a A), e classificar com base no conteúdo de uma ou mais colunas.

O Microsoft Excel classifica listas alfabeticamente como padrão. Se você deseja classificar meses e dias da semana de acordo com a ordem do calendário em vez da ordem alfabética, use uma ordem de classificação personalizada. Você também pode reordenar listas em uma ordem específica criando ordens de classificação personalizadas. Por exemplo, se você possui uma lista que contém a entrada “Baixo”, “Médio” ou “Alto” em uma coluna, pode-se criar uma ordem de classificação que ordene linhas que contêm “Baixo” primeiro, linhas que contêm “Médio” em seguida e linhas que contêm “Alto” por último.

2. Clique em uma célula da lista.
3. No menu *Dados*, clique em *Subtotais*.



4. Na caixa *A cada alteração em*, clique na coluna que contém os grupos para os quais você deseja subtotais. Esta deve ser a mesma coluna pela qual você classificou sua lista na etapa 1.
5. Na caixa *Usar função*, clique na função que você deseja usar para calcular os subtotais (ver *Funções de resumo para listas subtotalizadas*, mais adiante).
6. Na caixa *Adicionar subtotal a*, marque a caixa de seleção das colunas que contêm os valores para os quais você deseja subtotais.

Funções de resumo para listas subtotalizadas

Pode-se usar qualquer uma das funções abaixo para resumir os dados em uma lista. Clique na função desejada na caixa *Usar função* na caixa de diálogo *Subtotais* (menu *Dados*, comando *Subtotais*).

Use esta função	Para resumir
Soma	A soma dos valores em uma lista. Esta é a função padrão para dados numéricos.

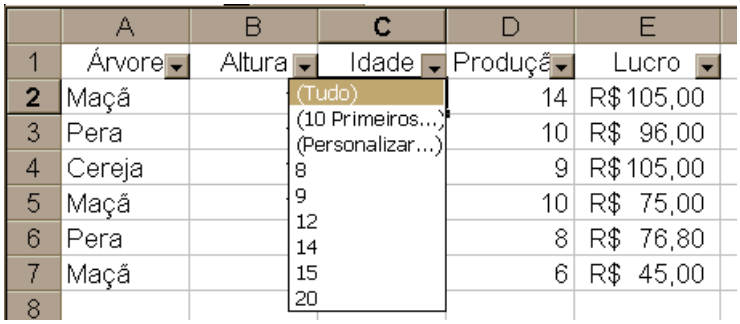
Contar	O número de itens em uma lista. Esta é a função padrão para dados não numéricos.
Média	A média dos valores em uma lista.
Máx	O maior valor em uma lista.
Mín	O menor valor em uma lista.
Produto	O resultado da multiplicação de todos os valores em uma lista.
Cont.Núms	O número de registros ou linhas em uma lista que contém dados numéricos.
DesvPad	Uma estimativa do desvio padrão de uma população, onde a lista é a amostra.
DesvPadp	O desvio padrão de uma população onde a lista é a população inteira.
Var	Uma estimativa da variância de uma população onde a lista é a amostra.
Varp	A variância de uma população onde a lista é a população inteira.

Famous Cells and Ranges

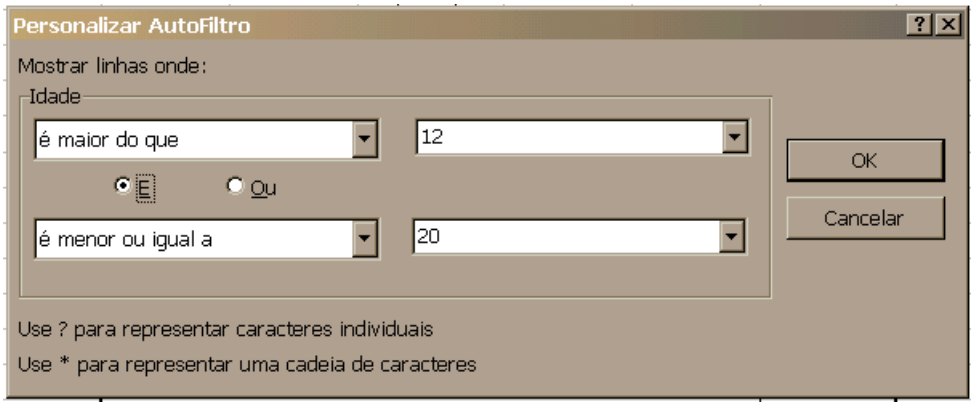
AH:HÁ	The discovery range
AM:FM	The radio range
BY:BY	The farewell range
HO:HO	The Santa Claus range
GO2	The destination cell
FU2	The same to you cell
F16	The fighter jet cell
IC2	The double-vision cell
U2	The Irish rock group cell
R2:D2	The android range
I1:U1	The tied game cell

Filtros

O filtro é uma maneira rápida de localizar subconjuntos de dados em uma lista. Para filtrar uma lista, clique em qualquer célula da lista, aponte para *Filtrar* no menu *Dados* e clique em *AutoFiltro*. O Microsoft Excel exibe setas à direita dos rótulos das colunas na lista. Para selecionar o valor



que deseja exibir na lista, clique na seta e, em seguida, clique no valor. Será exibida uma caixa contendo todos os valores contidos na lista, a partir do qual pode-se selecionar qual o valor a ser filtrado. A opção *Personalizar...* permite acrescentar alguma flexibilidade a filtragem, através de



um formulário onde podem ser especificados todas as cláusulas do filtro, como mostrado abaixo.

Filtro avançado

Para filtragens mais elaboradas pode-se fazer uso do Filtro Avançado. Este comando trabalho utilizando critérios que devem ser expressos na própria planilha, normalmente em uma região acima da lista que se deseja filtrar. Assim, o processo para se utilizar o filtro avançado é o seguinte:

1. Insira algumas linhas acima da lista com a qual se está trabalhando.
2. Copie, na primeira linha desta área, a primeira linha da lista (que contém os rótulos de cada coluna).

3. Insira, logo abaixo de cada rótulo, os critérios de pesquisa que devem ser utilizados. Critérios de vários campos na mesma linha funcionam como se estivessem conectados utilizando o operador E, enquanto critérios de vários campos em linhas diferentes funcionam como se estivessem conectados através do operador OR. Nos exemplos mostrados abaixo, a primeira situação pode ser expressa como: encontre todas as macieiras com idade > 12 e altura > 10 .

	A	B	C	D	E
1	Árvore	Altura	Idade	Produção	Lucro
2	Maçã		> 12	> 10	
3					
4	Árvore	Altura	Idade	Produção	Lucro
5	Maçã	18	20	14	R$ 105,00
6	Pera	12	12	10	R$ 96,00
7	Cereja	13	14	9	R$ 105,00
8	Maçã	14	15	10	R$ 75,00
9	Pera	9	8	8	R$ 76,80
10	Maçã	8	9	6	R$ 45,00
11					

Já no segundo caso, a expressão seria: encontre todas as árvores que ou são macieiras, ou possuem idade maior que 12 ou possuem altura > 10.

4. Ative o comando *Filtro avançado*, presente no submenu Filtrar do menu *Dados*. Será exibida uma caixa de diálogo solicitando as áreas onde estão localizadas a lista e os critérios. Uma

	A	B	C	D	E
1	Árvore	Altura	Idade	Produção	Lucro
2	Maçã				
3			> 12		
4				> 10	
5					
6	Árvore	Altura	Idade	Produção	Lucro
7	Maçã	18	20	14	R$ 105,00
8	Pera	12	12	10	R$ 96,00
9	Cereja	13	14	9	R$ 105,00
10	Maçã	14	15	10	R$ 75,00
11	Pera	9	8	8	R$ 76,80
12	Maçã	8	9	6	R$ 45,00
13					

vez identificadas estas áreas, pode-se clicar em *Ok* e o filtro será ativado. Deve-se notar que é possível filtrar a lista no mesmo local onde ela se encontra ou copiar os dados filtrados para outro local, o que pode ser bastante útil para a criação de planilhas contendo subconjuntos dos dados originais.

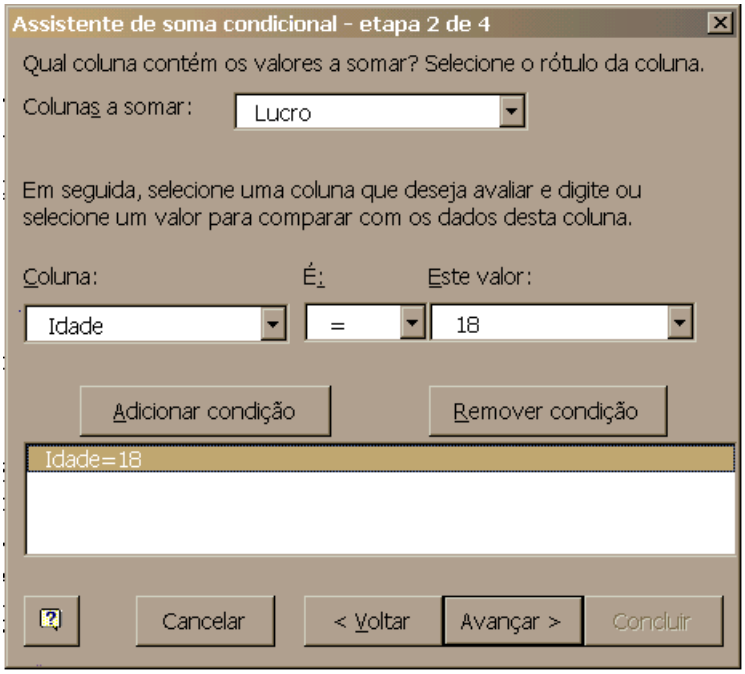
5. Quando o filtro avançado estiver em uso, será habilitado também o comando *Mostrar todos*, presente no submenu *Filtrar* do menu *Dados*. Este comando permitirá retornar a situação da planilha para o seu estado original, sem filtragem.

Usando o assistente de soma condicional

Se você deseja resumir valores em uma lista com base em condições específicas, pode-se usar o *Assistente de soma condicional*. Por exemplo, se sua lista contiver valores de vendas de

diferentes vendedores, o *Assistente de soma condicional* poderá ajudá-lo a criar uma fórmula que calcule o valor total das vendas de um vendedor.

O *Assistente de soma condicional* é um programa suplementar; se o comando *Soma condicional* não estiver no submenu *Assistente* do menu *Ferramentas*, será necessário instalar e carregar o programa suplementar.



1. Clique em uma célula na lista.
2. No menu Ferramentas, aponte para Assistente, e clique em Soma condicional.
3. Siga as instruções apresentadas no assistente.

Funções de banco de dados

Quando você precisar analisar se os valores contidos em uma lista atendem a uma condição específica, ou critérios, poderá usar uma função de planilha de banco de dados. Por exemplo, em uma lista que contém informações de vendas, pode-se contar todas as linhas ou registros em que as vendas sejam maiores que 1.000 mas menores que 2.500.

Algumas funções de planilha de gerenciamento de banco de dados e de listas têm nomes que começam com as letras "BD". Essas funções, conhecidas também como Bdfunções, têm três argumentos - *banco de dados*, *campo* e *critérios*.

- argumento de banco de dados é o intervalo que contém sua lista. Você deve incluir a linha que contém os rótulos de coluna no intervalo.
- argumento de campo é o rótulo para a coluna que você deseja resumir.
- argumento de critérios é o intervalo que contém uma condição especificada por você.

As tabelas a seguir exemplificam um banco de dados para um pequeno pomar. Cada registro contém informações sobre uma árvore. No Microsoft Excel podemos chamar o primeiro intervalo de Banco_dados e o segundo intervalo de Critérios.

Árvore	Altura	Idade	Produção	Lucro
Maçã	18	20	14	R$105,00
Pêra	12	12	10	R$ 96,00
Cereja	13	14	9	R$105,00
Maçã	14	15	10	R$ 75,00
Pêra	9	8	8	R$ 76,80
Maçã	8	9	6	R$ 45,00

Árvore	Altura	Idade	Produção	Lucro	Altura
Maçã	>10				<16
Pêra					

Vejamos os resultados obtidos pelas diferentes funções de banco de dados atuando nos dados acima:

BDCONTAR(Banco_dados;"Idade";A1:F2) é igual a 1. Esta função analisa os registros de macieiras com altura entre 10 e 16 e conta quantos campos Idade nestes registros contêm números.

BDCONTARA(Banco_dados;"Lucro";A1:F2) é igual a 1. Esta função analisa os registros de macieiras com altura entre 10 e 16 e conta quantos campos Lucro nesses registros não estão em branco.

BDMÁX(Banco_dados;"Lucro";A1:A3) é igual a R$ 105,00, o lucro máximo de macieiras e pereiras.

BDMÍN(Banco_dados;"Lucro";A1:B2) é igual a R$ 75,00, o lucro mínimo de macieiras acima de 10.

BDSOMA(Banco_dados;"Lucro";A1:A2) é igual a R$ 225,00, o lucro total de macieiras.

BDSOMA(Banco_dados;"Lucro";A1:F2) é igual a R$ 75,00, o lucro total de macieiras com uma altura entre 10 e 16.

BDMULTIPL(Banco_dados;"Produção";A1:F2) é igual a 140, o produto das produções das macieiras com altura entre 10 e 16.

BDMÉDIA(Banco_dados;" Produção";A1:B2) é igual a 12, a produção média das macieiras acima de 10 pés de altura.

BDMÉDIA(Banco_dados;3;Banco_dados) é igual a 13, a idade média de todas as árvores no banco de dados.

BDEST(Banco_dados;"Produção";A1:A3) é igual a 2,97, o desvio padrão estimado na produção das macieiras e pereiras se os dados do banco de dados forem apenas uma amostra da população total do pomar.

BDDESVPA(Banco_dados;"Produção";A1:A3) é igual a 2,65, o desvio padrão verdadeiro na produção das macieiras e pereiras se os dados do banco de dados representarem toda a população do pomar.

BDVAREST(Banco_dados;"Produção";A1:A3) é igual a 8,8, a variância estimada na produção das macieiras e pereiras se os dados no banco de dados forem apenas uma amostra da população total do pomar.

BDVARP(Banco_dados;"Produção";A1:A3) é igual a 7,04, a variância verdadeira na produção das macieiras e pereiras se os dados no banco de dados representarem toda a população do pomar.

BDEXTRAIR(Banco_dados;"Produção";Critérios) retornará o valor de erro #NÚM! porque mais de um registro coincide com os critérios.

Consolidar dados

Pode-se consolidar dados de quatro maneiras:

- Usando referências 3-D, o método preferido. Ao usar referências 3-D, não existem restrições quanto ao layout dos dados nas áreas de origem.
- Por posição, quando os dados em todas as áreas de origem são dispostos em ordem e localização idênticas; por exemplo, para combinar dados de uma seqüência de planilhas criadas a partir do mesmo modelo.

  Se você estiver inserindo dados, usando uma seqüência de formulários de planilha baseados no mesmo modelo, e desejar listar os dados inseridos em cada formulário em uma única planilha, experimente usar o Assistente de modelo com rastreamento de dados.
- Por categoria, quando você deseja resumir um conjunto de planilhas que têm os mesmos rótulos mas organizam os dados de modo diferente. Este método combina os dados que têm rótulos correspondentes em cada planilha.
- Criando uma tabela dinâmica. Este método é semelhante à consolidação por categoria, mas oferece maior flexibilidade para a reorganização das categorias.

Diretrizes para especificar áreas de origem para uma consolidação

Ao consolidar dados, você especifica as áreas de origem dos dados, quer em fórmulas 3-D, quer na caixa *Referência* da caixa de diálogo *Consolidar*. As áreas de origem podem ser intervalos de células na mesma planilha que a tabela de consolidação, em diferentes planilhas na mesma pasta de trabalho ou em diferentes pastas de trabalho.

Use as diretrizes a seguir para definir áreas de origem.

- Para facilitar ainda mais o rastreamento das áreas de origem, nomeie cada intervalo e use os nomes na caixa Referência.
- Quando as áreas de origem e a área de destino estiverem na mesma planilha, use as referências de células ou intervalos de células, ou seus nomes.
- Quando as origens e o destino estiverem em planilhas diferentes, use as referências ou os nomes das planilhas e células ou intervalos de células. Por exemplo, para incluir um intervalo chamado Orçamento, contido na planilha Marketing em sua pasta de trabalho, insira Marketing!Orçamento.
- Quando as origens e o destino estiverem em pastas de trabalho diferentes, use as referências ou os nomes de pastas, planilhas, e células ou intervalos. Por exemplo, para incluir um intervalo chamado Vendas, contido em uma planilha chamada Região Leste existente em outra pasta de trabalho chamada 1996, na mesma pasta, digite o seguinte:

  '[1996.xls]RegiãoLeste'!Vendas
- Quando as origens e o destino estiverem em diferentes pastas de trabalho alocadas em diferentes localizações, use as referências ou nomes de caminho completo, de pastas, planilhas, células ou intervalos. Por exemplo, para incluir um intervalo chamado Receita, contido na planilha Fevereiro, existente na pasta de trabalho Departamento de Vendas, na pasta Planilhas de Orçamentos, digite o seguinte:

  '[C:\Planilhas de Orçamento:\Departamento de Vendas.xls]Fevereiro'!Receita

Observação: Pode-se omitir os nomes de planilhas nas referências, se os intervalos tiverem nomes atribuídos por você, em vez de nomes criados pelo Microsoft Excel. Por exemplo, '[1996.xls]'!Vendas ou '[C:\Planilhas de Orçamento\Departamento de Vendas.xls]'!Receita.

Dica: Para inserir uma referência de origem sem digitar, clique na caixa *Referência* e, em seguida, selecione a área de origem. Para selecionar uma área de origem em outra pasta de trabalho, clique em *Procurar*. Para remover temporariamente a caixa de diálogo *Consolidar* enquanto você seleciona a área de origem, clique em *Recolher caixa de diálogo*.

Consolidar dados usando referências 3-D

- Na planilha de consolidação, copie ou insira os rótulos que você deseja para os dados consolidados.
- Clique em uma célula que você deseja que contenha dados consolidados.
- Digite uma fórmula que inclua referências às células de origem em cada planilha que contém os dados que você deseja consolidar.
- Repita as etapas 2 e 3 para cada célula em que você deseja consolidar dados.

Dica: Para inserir uma referência em uma fórmula sem digitar, insira a fórmula até o ponto em que você precisa da referência e, em seguida, clique na célula na planilha. Se a célula estiver em outra planilha, primeiro clique na guia da planilha e, em seguida, na célula.

Consolidar dados por posição

1. Clique na célula superior esquerda da área de destino para os dados consolidados.
2. No menu *Dados*, clique em *Consolidar*.
3. Na caixa *Função*, clique na função de resumo que você deseja que o Microsoft Excel use para consolidar os dados.
4. Na caixa *Referência*, insira uma área de origem que você deseja consolidar.
5. Clique em *Adicionar*.
6. Repita as etapas 4 e 5 para cada área de origem a ser consolidada.
7. Para atualizar automaticamente a tabela de consolidação quando os dados fonte forem alterados, marque a caixa de seleção *Criar vínculos com os dados de origem*.

Para criar vínculos, as áreas de origem e de destino devem estar em planilhas diferentes. Assim que você criar os vínculos, não poderá adicionar novas áreas de origem nem alterar as áreas de origem incluídas na consolidação.

Observação: Quando você consolida por posição, o Microsoft Excel não copia os rótulos de categorias existentes nas áreas de origem para a área de destino. Se você desejar rótulos na planilha de destino, copie ou insira-os manualmente.

Consolidar dados por categoria

1. Clique na célula superior esquerda da área de destino para os dados consolidados.
2. No menu *Dados*, clique em *Consolidar*.
3. Na caixa *Função*, clique na função de resumo que você deseja que o Microsoft Excel use para consolidar os dados.
4. Na caixa *Referência*, insira uma área de origem que você deseja consolidar. Certifique-se de incluir os rótulos de dados na seleção.
5. Clique em *Adicionar*.

6. Repita as etapas 4 e 5 para cada área de origem que você deseja consolidar.
7. Em *Usar rótulos na*, marque as caixas de seleção que indicam onde os rótulos estão posicionados na área de origem: quer na linha superior, quer na coluna esquerda ou em ambas.
8. Para atualizar automaticamente a tabela de consolidação quando os dados fonte forem alterados, marque a caixa de seleção *Criar vínculos com os dados de origem*.

Para criar vínculos, as áreas de origem e de destino devem estar em planilhas diferentes. Assim que você criar vínculos, não poderá adicionar novas áreas de origem nem alterar as áreas de origem incluídas na consolidação.

Observação: Os rótulos existentes em uma área de origem, que não coincidirem com os rótulos em outras áreas de origem, resultarão em linhas ou colunas separadas quando você consolidar dados.

Criar uma Tabela dinâmica

Tabela dinâmica é uma tabela interativa que resume e analisa dados de listas e tabelas existentes. Use o *Assistente da tabela dinâmica* para especificar a lista ou tabela que você deseja usar e definir como você quer organizar os dados na Tabela dinâmica. Após ter criado uma Tabela dinâmica, pode-se reorganizar os dados arrastando os campos e itens.

Quando você baseia uma Tabela dinâmica em dados externos, é possível que você queira recuperar os dados externos antes de criar a Tabela dinâmica.

1. Abra a pasta de trabalho onde você deseja criar a Tabela dinâmica.

   Se você estiver baseando a Tabela dinâmica em uma lista ou banco de dados do Microsoft Excel, clique em uma célula da lista ou do banco de dados.
2. No menu Dados, clique em Relatório da tabela dinâmica.
3. Siga as instruções do Assistente da tabela dinâmica.

Assistente de modelo com rastreamento de dados

O suplemento *Assistente de modelo com rastreamento de dados* cria um modelo que vincula células selecionadas em uma pasta de trabalho a campos em um banco de dados.

Quando você baseia uma nova pasta de trabalho no modelo e insere dados nas células vinculadas, o Microsoft Excel cria um novo registro no banco de dados e copia os dados para os campos de dados correspondentes.

Por exemplo, suponha que você deseje rastrear acidentes de trabalho de várias fábricas conectadas por uma rede. Primeiro, crie um formulário em uma planilha que solicite as informações necessárias para cada relatório de acidente. Em seguida, use o *Assistente de modelo* para criar um modelo a partir do formulário e vincular o modelo a um banco de dados central. Quando os trabalhadores preencherem um novo relatório de acidente de trabalho baseado no modelo, criarão um relatório “impresso” e um registro correspondente no banco de dados.

Se as pastas de trabalho já existentes contiverem dados que você deseja incluir no novo banco de dados, o Assistente de modelos poderá adicionar os dados automaticamente. Entretanto, as posições dos dados nas pastas de trabalho existentes deverão coincidir com as posições das células que você selecionar para os campos de dados. Por exemplo, se na etapa 3 do Assistente de modelo você indicar que o campo ID do Funcionário está inserido na célula $D$4 da planilha chamada Dados_de_Pessoal, todas as pastas de trabalho existentes deverão conter o número do funcionário na célula Dados_de_Pessoal!$D$4. Se as posições dos dados não coincidirem, é possível que sejam adicionados registros incompletos ao banco de dados.

O banco de dados pode ser uma lista do Microsoft Excel ou um banco de dados do Microsoft Access, Microsoft FoxPro, dBase ou Paradox para o qual você instalou o driver ODBC necessário e outros componentes de acesso a dados. Armazene o banco de dados em um local da rede que todos os usuários do formulário possam acessar.

Pode-se criar relatórios a partir dos dados gravados ou usá-los como você usaria qualquer banco de dados. Para alterar os dados contidos em um registro, pode-se reabrir e editar a cópia gravada do formulário associada a esse registro. Se você excluir a cópia do formulário, o registro correspondente será mantido no banco de dados. Pode-se excluir o registro como o faria com qualquer registro nesse tipo de banco de dados.

Para tornar o modelo disponível para todos os usuários, armazene o modelo em uma posição acessível da rede. Em seguida, crie um atalho para o modelo e instrua os usuários a copiar o atalho para a pasta *Modelos* existente na área de trabalho de cada um. A pasta *Modelos* está na pasta em que o Office ou o Microsoft Excel foi instalado.

Para obter maiores informações sobre como usar o Assistente de modelo, clique em .

Criar um modelo de formulário que copia dados da planilha para um banco de dados

Se o comando *Assistente de modelo* não estiver no menu *Dados*, você precisará instalar o suplemento *Assistente de modelo com rastreamento de dados* antes de executar este procedimento.

1. Abra a pasta de trabalho em que você inserirá os dados que deseja copiar para um banco de dados.

   Se você já salvou um formulário como um modelo de pasta de trabalho, clique em Novo no menu Arquivo e crie uma nova pasta de trabalho a partir do modelo. Clique em Salvar, e salve a nova pasta de trabalho com outro nome. Em seguida, inicie o Assistente de modelo, o que salvará a pasta de trabalho como um modelo.
2. Insira rótulos para os dados que você deseja que sejam inseridos na planilha.
   Insira cada rótulo em uma célula acima ou à esquerda da célula que contém os dados. Os rótulos serão usados como nomes de campo no banco de dados.
3. No menu Dados, clique em Assistente de modelo.
4. Siga as etapas do assistente.

Observação O banco de dados que você vincula ao modelo de formulário pode ser uma lista do Microsoft Excel ou um banco de dados do Microsoft Access, Microsoft FoxPro, dBase ou Paradox para o qual você instalou o driver ODBC necessário e outros componentes de acesso a dados.

=OR(B2,NOT(B2))

A fórmula favorita de Shakespeare

É um fato bem pouco conhecido, mas Shakespeare estava trabalhando em uma planilha Excel quando criou uma fórmula que lhe inspirou um de seus mais famosos trabalhos...

Menus de aplicação

Um fator essencial na criação de aplicativos personalizados é oferecer uma forma simples e consistente para o usuário interagir com seu aplicativo – menus personalizados e barra de ferramentas customizadas.

A interface com o usuário

Menus são listas suspensas, a partir das quais o usuário pode fazer suas escolhas de comandos. Estes comandos podem ser agrupados em um mesmo nível de menu fornecendo um método simples e consistente. Podemos também atribuir teclas de atalho aos comandos mais usuais, agilizando o manuseio da aplicação.

Além dos menus podemos customizar as barras de ferramentas através dos seus botões, mas este método somente será acessível através do mouse. Como vantagem, as barras de ferramentas permanecem visíveis todo o tempo dispensando a necessidade de treinamentos exaustivos.

Devemos fornecer um conjunto de opções que se complementem permitindo ao usuário escolher a melhor forma de acesso aos comandos. Eventualmente podemos prescindir das duas opções acima e colocar comandos diretamente sobre a planilha de trabalho, como vimos anteriormente.

O sistema de menu

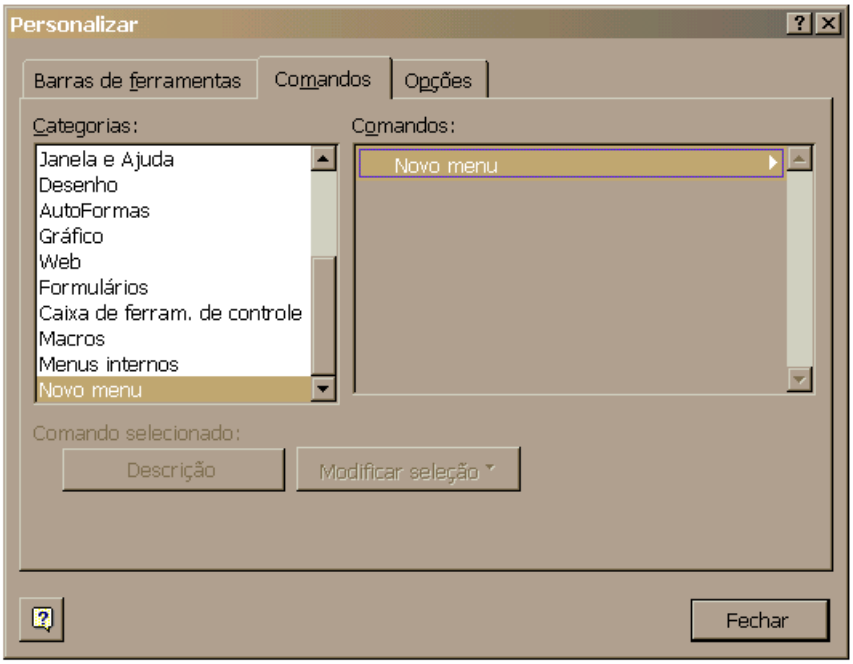
O sistema de menus do Microsoft Excel é composto de um conjunto inteiro de menus disponíveis de dos itens de cada menu. Por padrão, o Excel exibia a barra de menus sensível ao contexto do usuário, ou seja, opções que não podem ser ativadas ficam em tom cinza.

Podemos abrir um novo comando na barra de menus existente no Excel ou podemos criar um novo botão de menu em uma barra de ferramentas. Antes de mais nada é necessário um planejamento sobre como será o menu.

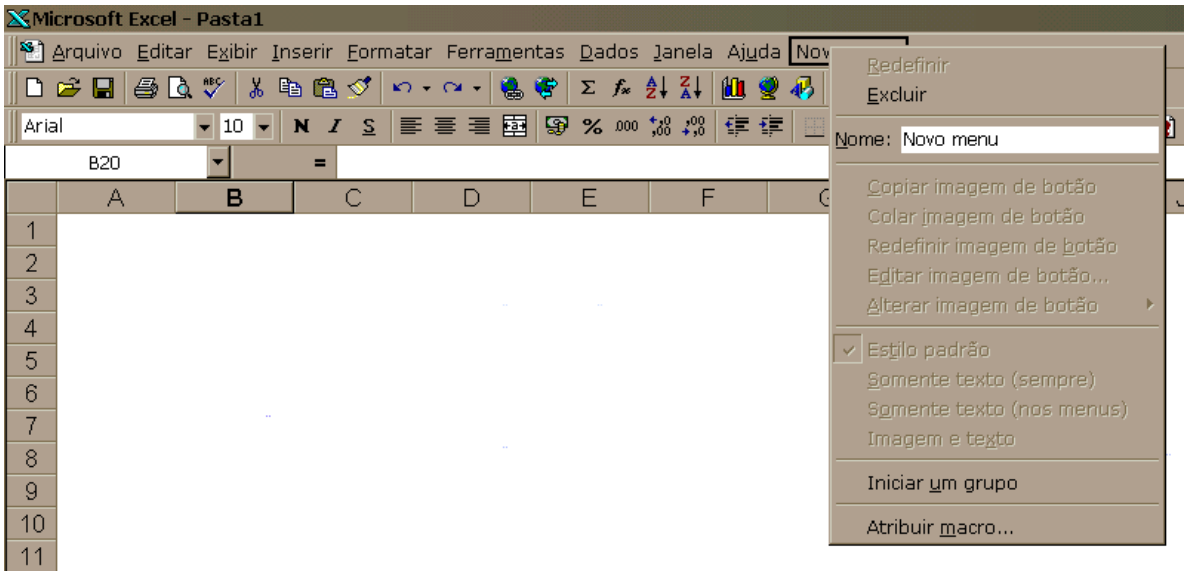
Criando um novo item de menu dentro do menu padrão

Para adicionar um comando a um menu execute os seguintes passos:

1. Mostre a barra de menus que contém o menu ao qual você deseja adicionar um comando. Normalmente este será o menu padrão da aplicação.
2. No menu Ferramentas, clique em Personalizar e depois clique na guia Comandos.
3. Na caixa Categorias, clique na categoria do comando “Novo Menu”



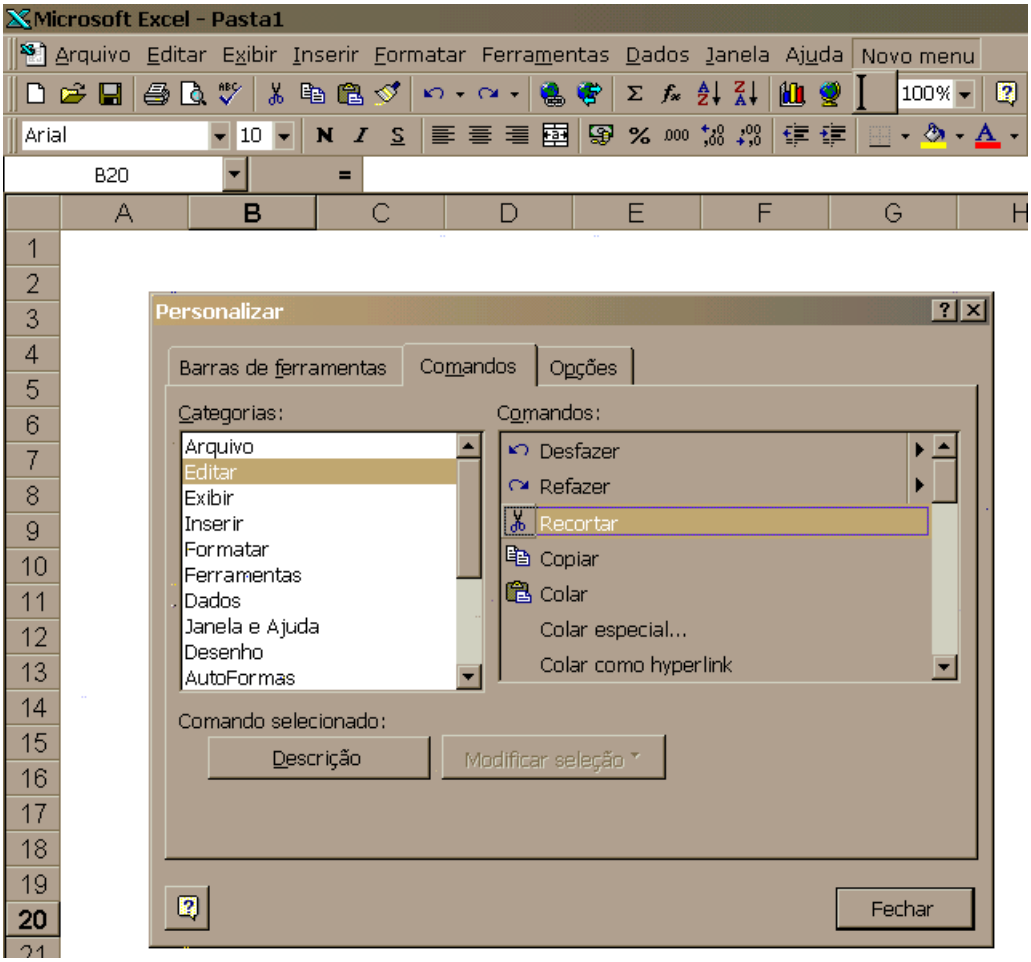
4. Arraste o comando “Novo Menu” da caixa Comandos para o menu na barra de menus soltando-a lá. Observe que uma barra escura mostra a posição onde o novo comando será



inserido. Assim que o mouse for liberado aparecerá no local o novo menu.

5. Modifique a propriedade “Nome” e, em seguida, escolha uma macro a ser atribuída ao menu. Nem todas as propriedades se aplicam a este menu.
6. Outros comandos podem ser inseridos dentro deste menu, bastando que eles sejam arrastados para a caixa existente abaixo do menu.

Na figura, o comando de menu “Recortar” está sendo movido para o primeiro nível dentro do novo comando de menu.

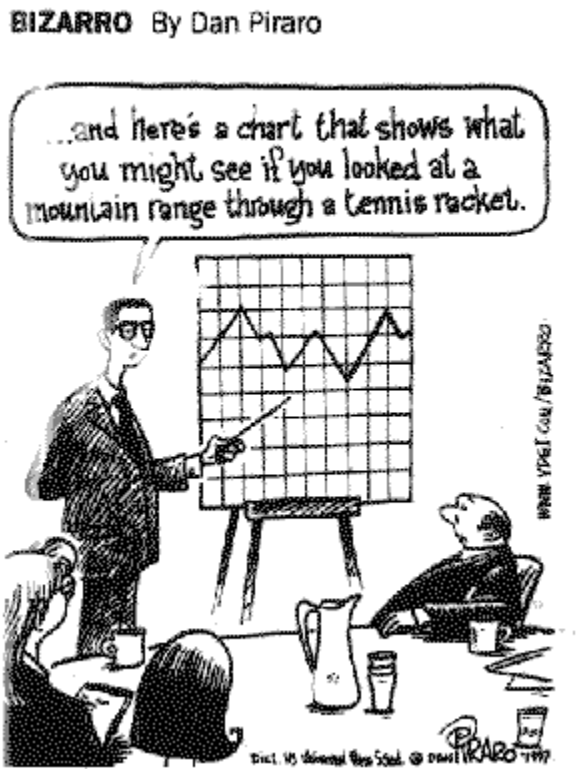


Criando um novo botão na barra de ferramentas

Para adicionar um menu personalizado a uma barra de ferramentas o procedimento é o mesmo adotado para criar menus. Mas o programador deverá soltar o mouse com o comando desejado sobre uma das barras de ferramentas existentes e não sobre a barra de menus.

1. Mostre a barra de ferramentas à qual você deseja adicionar um menu personalizado.
2. No menu Ferramentas, clique em Personalizar e depois clique na guia Comandos
3. Na caixa Categorias, clique em Novo menu
4. Arraste Novo menu da caixa Comandos para a barra de ferramentas exibida.
5. Clique com o botão direito do mouse no novo menu na barra de ferramentas e, em seguida, digite um nome na caixa Nome no menu de atalho. Pressione a tecla Enter.
6. Para adicionar um comando ao menu personalizado, clique no menu personalizado na barra de ferramentas para exibir uma caixa vazia. Clique na categoria do comando na caixa Categorias e, em seguida, arraste o comando da caixa Comandos para a caixa vazia no menu personalizado.

Ferramentas estatísticas



O Microsoft Excel fornece um conjunto de ferramentas para análise de dados, denominado Ferramentas de análise, que pode ser usado para pular etapas no desenvolvimento de análises estatísticas ou de engenharia complexas. Você fornece os dados e os parâmetros para cada análise; a ferramenta utiliza as funções de macro de estatística ou engenharia adequadas e exibe os resultados em uma tabela de saída. Algumas ferramentas geram gráficos além das tabelas de saída.

Para exibir uma lista das ferramentas de análise disponíveis, clique em *Análise de dados* no menu *Ferramentas*. Se o comando *Análise de dados* não estiver no menu *Ferramentas*, execute o Programa de Instalação para instalar as Ferramentas de análise. Depois de instalar as *Ferramentas de análise*, você deve selecioná-la no *Gerenciador de suplementos*.

Para utilizar estas ferramentas, você precisa estar familiarizado com a área de estatística ou engenharia específica para a qual deseja desenvolver análises.

Observação: O Microsoft Excel fornece muitas outras funções estatísticas, financeiras e de engenharia para as planilhas. Para visualizar uma lista das funções de planilha disponíveis, clique em *Editar fórmula* na barra de fórmulas e, em seguida, clique na seta abaixo em inserção de função .

Instalar e usar as Ferramentas de análise

Para usar uma ferramenta de análise, você precisa organizar os dados que deseja analisar em colunas ou linhas na sua planilha. Este é o seu intervalo de entrada.

Se o comando Análise de dados não estiver no menu Ferramentas, você precisa instalar as Ferramentas de análise no Microsoft Excel.

Para instalar as Ferramentas de análise

1. No menu Ferramentas, clique em Suplementos.

   Se Ferramentas de análise não estiver incluído na caixa de diálogo Suplementos, clique em Procurar e localize a unidade de disco, o nome da pasta e o nome do arquivo para o suplemento Ferramentas de análise, Analys32.xll - geralmente localizado na pasta Biblioteca\Análise - ou execute o Programa de Instalação se ele não estiver instalado.
2. Marque a caixa de seleção Ferramentas de análise.

Para usar as Ferramentas de análise

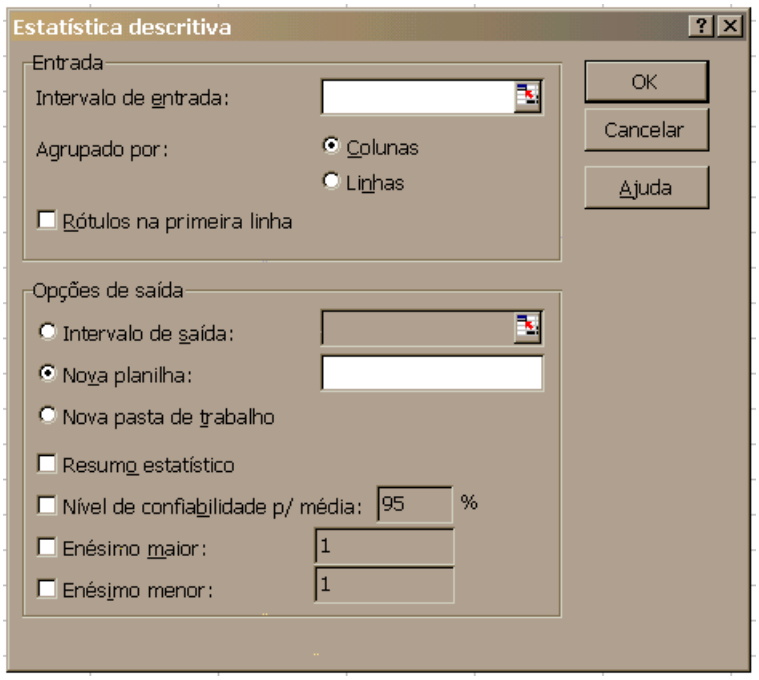
1. No menu Ferramentas, clique em Análise de dados.
2. Na caixa Ferramentas de análise, clique na ferramenta que você deseja utilizar.
3. Insira o intervalo de entrada e o intervalo de saída e, em seguida, selecione as opções desejadas.

Observação Os suplementos selecionados na caixa de diálogo Suplementos permanecem ativos até que você os remova.

Ferramenta de análise Estatística Descritiva

Esta ferramenta de análise gera um relatório de estatística monovariável dos dados no intervalo de entrada, fornecendo informações sobre a tendência central e variabilidade dos seus dados.

Caixa de diálogo Estatística Descritiva



Intervalo de entrada

Insira a referência de célula para os dados que você deseja analisar. A referência deve consistir em dois ou mais intervalos de dados adjacentes, ordenados em colunas ou linhas.

Agrupado por

Para indicar se os dados no intervalo de entrada estão ordenados em linhas ou em colunas, clique em Linhas ou Colunas.

Rótulos na primeira linha/Rótulos na primeira coluna

Se a primeira linha do seu intervalo de entrada contiver rótulos, marque a caixa de seleção Rótulos na primeira linha. Se os rótulos estiverem na primeira coluna do intervalo de entrada, marque a caixa de seleção Rótulos na primeira coluna. Desmarque a caixa se o intervalo de entrada não contiver rótulos; o Microsoft Excel gera os rótulos de dados adequados para a tabela de saída.

Nível de confiabilidade p/média

Selecione esta opção se quiser incluir uma linha na tabela de saída para o nível de confiança da média. Na caixa, insira o nível de confiança a ser usado. Por exemplo, um valor de 95% calcula o nível de confiança da média a uma significância de 5%.

Enésimo maior

Selecione esta opção se quiser incluir uma linha na tabela de saída para o enésimo maior valor para cada intervalo de dados. Na caixa, insira o número a ser usado para N. Se você inserir 1, esta linha conterá o máximo do conjunto de dados.

Enésimo menor

Selecione esta opção se quiser incluir uma linha na tabela de saída para o enésimo menor valor para cada intervalo de dados. Na caixa, insira o número a ser usado para N. Se você inserir 1, esta linha conterá o mínimo do conjunto de dados.

Intervalo de saída

Insira a referência para a célula superior esquerda da tabela de saída. Esta ferramenta gera duas colunas de informações para cada conjunto de dados. A coluna esquerda contém rótulos de estatísticas e a coluna direita contém as estatísticas. O Microsoft Excel cria uma tabela de duas colunas de estatísticas para cada coluna ou linha no intervalo de entrada, dependendo da opção Agrupado por selecionada.

Nova planilha

Clique nesta opção para inserir uma nova planilha na pasta de trabalho atual e colar os resultados na célula A1 da nova planilha. Para nomear a nova planilha, digite um nome na caixa.

Nova pasta de trabalho

Clique nesta opção para criar uma nova pasta de trabalho e colar os resultados em uma nova planilha na nova pasta de trabalho.

Resumo estatístico

Selecione esta opção se quiser que o Microsoft Excel gere um campo para cada uma das seguintes estatísticas na tabela de saída: Média, Erro padrão (da média), Mediana, Modo, Desvio padrão, Variância, Curtose, Distorção, Intervalo, Mínimo, Máximo, Soma, Contagem, Maior (n), Menor (n) e Nível de confiança.

Ferramenta de análise Histograma

Esta ferramenta de análise calcula as freqüências individuais e cumulativas para um intervalo de células de dados e dados binários. Esta ferramenta gera dados para o número de ocorrências de um valor em um conjunto de dados. Por exemplo, em uma turma de 20 alunos, você poderia definir a distribuição dos resultados das notas nas categorias de graus por letras. Uma tabela do histograma apresenta os limites do grau em letras e o número de pontos entre o limite mais baixo e o limite atual. O resultado único mais freqüente é a moda dos dados.

Caixa de diálogo Histograma

Intervalo de entrada

Insira a referência para o intervalo de dados que você deseja analisar.

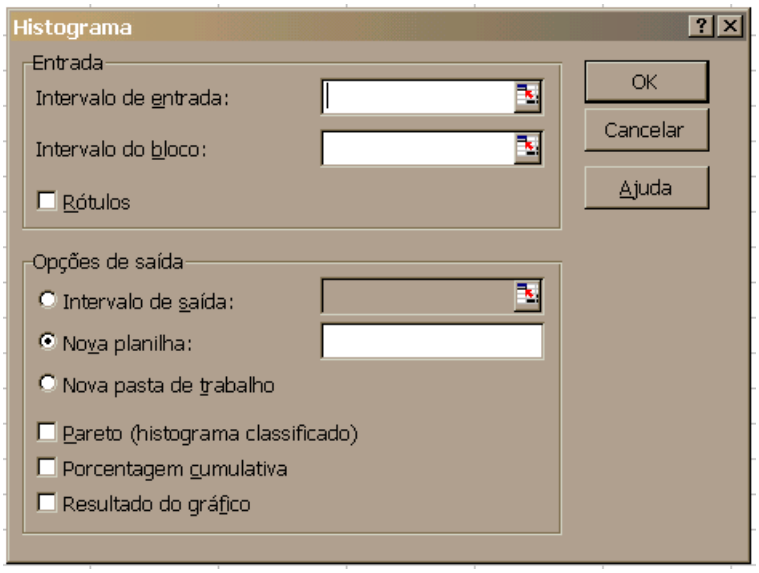
Intervalo do bloco (opcional)

Insira a referência de célula a um intervalo com um conjunto opcional de valores limites que definem os intervalos binários. Estes valores devem estar em ordem crescente. O Microsoft Excel conta o número de pontos de dados entre o número binário atual e o binário adjacente mais alto, se houver. Conta-se um número em um binário específico se este for igual ou menor do que o número binário até o último binário. Todos os valores abaixo do valor binário e acima do valor do último binário são contados.

Se você omitir o intervalo binário, o Microsoft Excel criará um conjunto de números binários uniformemente distribuídos entre os valores mínimo e máximo dos dados.

Rótulos

Selecione esta opção se a primeira linha ou coluna do seu intervalo de entrada contiver rótulos. Desmarque esta opção se o intervalo de entrada não contiver rótulos; o Microsoft Excel gera os rótulos de dados adequados para a tabela de saída.



Intervalo de saída

Insira a referência da célula superior esquerda da tabela de saída. O Microsoft Excel determinará automaticamente o tamanho da área de saída e exibirá uma mensagem se a tabela de saída estiver prestes a substituir os dados existentes.

Nova planilha

Clique nesta opção para inserir uma nova planilha na pasta de trabalho atual e colar os resultados na célula A1 da nova planilha. Para nomear a nova planilha, digite um nome na caixa.

Nova pasta de trabalho

Clique nesta opção parar criar uma nova pasta de trabalho e colar os resultados em uma nova planilha na nova pasta de trabalho.

Pareto (histograma classificado)

Selecione esta opção para apresentar dados na tabela de saída em ordem de freqüência decrescente. Se esta opção não estiver selecionada, o Microsoft Excel apresentará os dados em ordem crescente e omitirá as três colunas da extrema direita que contêm os dados classificados.

Porcentagem cumulativa

Selecione esta opção para gerar uma coluna na tabela de saída para porcentagens cumulativas e incluir uma linha de porcentagens cumulativas no gráfico de histograma. Desmarque esta opção para omitir as porcentagens cumulativas.

Resultado do gráfico

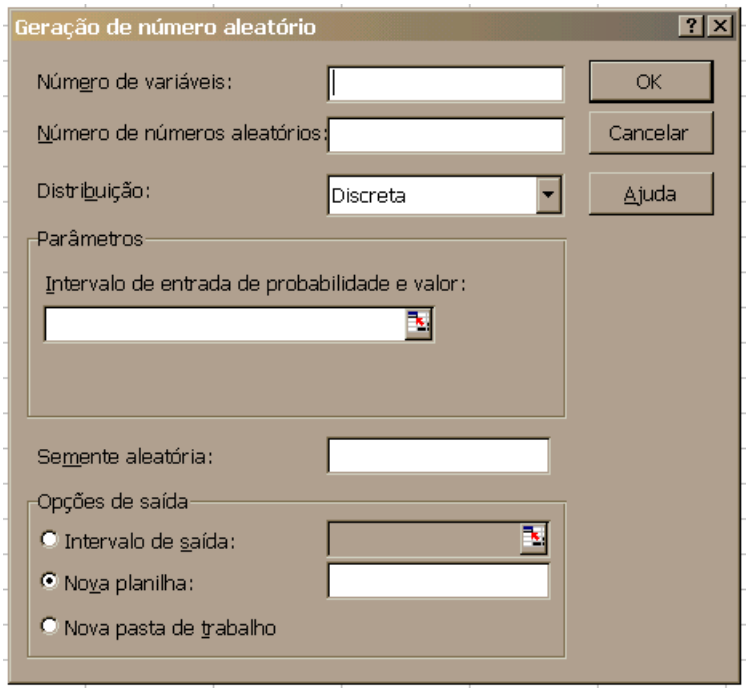
Selecione esta opção para gerar automaticamente um gráfico de histograma interno na tabela de saída.

Ferramenta de análise Geração de Número Aleatório

Esta ferramenta de análise preenche um intervalo com números randômicos independentes retirados de uma dentre várias distribuições. Pode-se usar essa ferramenta para caracterizar indivíduos em uma população com uma distribuição de probabilidade. Por exemplo, pode-se usar uma distribuição normal para caracterizar a população de alturas dos indivíduos ou pode usar uma distribuição de Bernoulli de dois resultados possíveis para caracterizar a população de resultados de “cara ou coroa”.

Observação: Para retornar um número randômico uniformemente distribuído maior ou igual a 0 e menor do que 1 sempre que a planilha for calculada, use a função RANDÔMICO(). Para retornar um número randômico entre os números especificados sempre que a planilha for calculada, use a função RANDÔMICOENTRE ().

Caixa de diálogo Geração de Número Aleatório



Número de variáveis

Insira o número de colunas de valores que você deseja na tabela de saída. Se não inserir um número, o Microsoft Excel preencherá as colunas da tabela de saída especificada.

Número de números aleatórios

Insira o número de pontos de dados que você deseja visualizar. Cada ponto de dados aparece em uma linha da tabela de saída. Se você não inserir um número, o Microsoft Excel preencherá todas as linhas da tabela de saída especificada.

Distribuição

Selecione o método de distribuição que você deseja usar para criar valores randômicos.

Uniforme

Caracterizada por limites inferiores e superiores. As variáveis são tiradas com probabilidade igual de todos os valores no intervalo. Um aplicativo comum usa uma distribuição uniforme no intervalo de 0 a 1.

Normal

Caracterizada por uma média e um desvio padrão. Um aplicativo comum usa uma média de 0 e um desvio padrão de 1 para a distribuição normal padrão.

Bernoulli

Caracterizada por uma probabilidade de êxito (valor p) em uma determinada tentativa. As variáveis randômicas Bernoulli têm o valor 0 ou 1. Por exemplo, pode-se formular uma variável randômica uniforme no intervalo de 0 a 1. Se a variável for menor ou igual à probabilidade de êxito, atribui-se o valor 1 à variável randômica Bernoulli; caso contrário, será atribuído o valor 0.

Binomial

Caracterizada por uma probabilidade de êxito (valor p) para várias tentativas. Por exemplo, pode-se gerar variáveis randômicas Bernoulli do número de tentativas, cuja soma é uma variável randômica binomial.

Poisson

Caracterizada por um valor lambda equivalente a 1/média. A distribuição Poisson é usada com freqüência para caracterizar o número de eventos que ocorre por unidade de tempo - por exemplo, uma taxa média na qual os carros chegam a um posto de pedágio.

Padronizada

Caracterizada por um limite inferior e superior, uma etapa, uma taxa de repetição para os valores e uma taxa de repetição para a seqüência.

Discreta

Caracterizada por um valor e pelo intervalo de probabilidade associado. O intervalo deve conter duas colunas: a coluna da esquerda contém valores e a da direita deve conter probabilidades associadas ao valor nesta linha. A soma das probabilidades deve ser 1.

Parâmetros

Insira um valor ou valores para caracterizar a distribuição selecionada.

Semente aleatória

Insira um valor opcional a partir do qual números randômicos possam ser gerados. Pode-se voltar a usar este valor para produzir os mesmos números randômicos posteriormente.

Intervalo de saída

Insira a referência da célula superior esquerda da tabela de saída. O Microsoft Excel determinará automaticamente o tamanho da área de saída e exibirá uma mensagem se a tabela de saída estiver prestes a substituir os dados existentes.

Nova planilha

Clique nesta opção para inserir uma nova planilha na pasta de trabalho atual e colar os resultados na célula A1 da nova planilha. Para nomear a nova planilha, digite um nome na caixa.

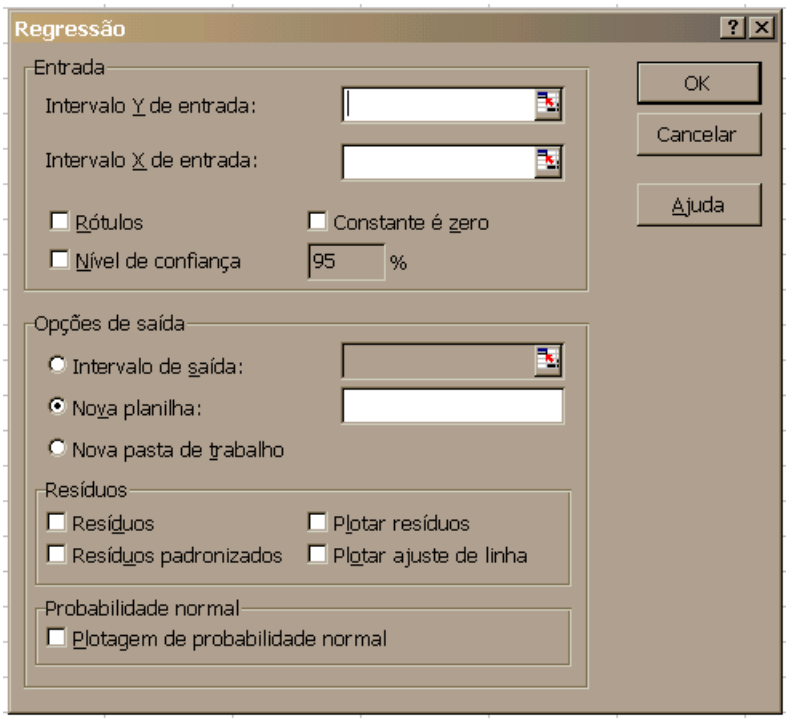
Nova pasta de trabalho

Clique nesta opção parar criar uma nova pasta de trabalho e colar os resultados em uma nova planilha na nova pasta de trabalho.

Ferramenta de análise Regressão

Esta ferramenta de análise executa uma análise de regressão linear, usando o método dos "mínimos quadrados" para ajustar uma linha através de um conjunto de observações. Pode-se usar esta ferramenta para analisar como uma variável dependente única é afetada pelos valores de uma ou mais variáveis independentes - por exemplo, como o desempenho de um atleta é afetado por fatores como idade, altura e peso. Pode-se ratear porções na medida do desempenho para cada um desses três fatores, com base em um conjunto de dados de desempenho e, em seguida, usar os resultados para prever o desempenho de um atleta novo ainda não testado.

Caixa de diálogo Regressão



Intervalo Y de entrada

Insira a referência para o intervalo de dados dependentes. O intervalo deve consistir em uma única coluna de dados.

Intervalo X de entrada

Insira a referência para o intervalo de dados independentes. O Microsoft Excel ordena variáveis independentes deste intervalo em ordem crescente da esquerda para a direita. O número máximo de variáveis independentes é 16.

Rótulos

Selecione esta opção se a primeira linha ou coluna do intervalo ou intervalos de entrada contiver rótulos. Desmarque esta opção se o intervalo de entrada não contiver rótulos; o Microsoft Excel gera os rótulos de dados adequados para a tabela de saída.

Nível de confiança

Selecione esta opção para incluir um nível adicional na tabela de resumo de saída. Nesta caixa, insira um nível de confiança adicional que deseja aplicar ao nível padrão de 95%.

Constante é zero

Selecione esta opção para forçar a linha de regressão a passar pela origem.

Intervalo de saída

Insira a referência para a célula superior esquerda da tabela de saída. Deixe pelo menos sete colunas para a tabela de resumo de saída, que inclui uma tabela anova, coeficientes, erro padrão de estimativa dos y, valores r2, número de observações e erro padrão dos coeficientes.

Nova planilha

Clique nesta opção para inserir uma nova planilha na pasta de trabalho atual e colar os resultados na célula A1 da nova planilha. Para nomear a nova planilha, digite um nome na caixa.

Nova pasta de trabalho

Clique nesta opção parar criar uma nova pasta de trabalho e colar os resultados em uma nova planilha na nova pasta de trabalho.

Resíduos

Selecione esta opção para incluir resíduos na tabela de saída de resíduos.

Resíduos padronizados

Selecione esta opção para incluir resíduos padronizados na tabela de saída de resíduos.

Plotar resíduos

Selecione esta opção para gerar um gráfico para cada variável independente por resíduo.

Plotar ajuste de linha

Selecione esta opção para gerar um gráfico para valores previstos por valores observados.

Plotagem de probabilidade normal

Selecione esta opção para gerar um gráfico usando plotagens de probabilidade normal.

Macros

O gênio da lâmpada

Um homem estava caminhando ao longo da praia quando encontrou uma lâmpada. Após esfrega-la apareceu um gênio que declarou: “Eu sou o gênio mais poderoso do mundo. Como sou tão poderoso, eu posso realizar qualquer desejo que você queira, mas apenas um.”

O homem mostrou um mapa do Oriente Médio e disse “Eu gostaria que existisse paz entre os povos desta área.”. O gênio respondeu: “Difícil... Estes povos tem estado em guerra desde o início dos tempos. Eles sempre estarão lutando e não há nada que eu possa fazer a respeito disso. Esta além de meus limites. Escolha outra coisa...”

O homem pensou um pouco e disse: “Bom, meu pessoal está mudando para Excel. Acho que você poderia ensinar-lhes um poucos deste tal de VBA...”

Gênio: “Uhhh, vamos dar uma olhada naquele mapa novamente...”

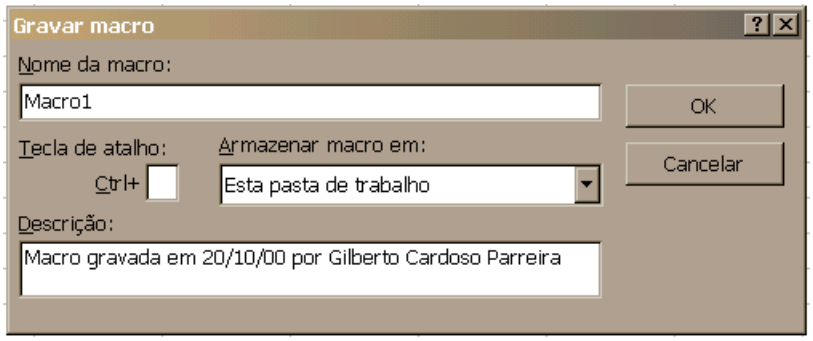
Caso você execute uma tarefa várias vezes no Microsoft Excel, é possível automatizá-la com uma macro. Uma macro é uma seqüência de comandos e funções armazenadas em um módulo do Visual Basic e pode ser executada sempre que você precisar executar a tarefa. A macro é gravada da mesma forma que uma música em um toca-fitas. Em seguida, você executa a macro para repetir, ou “reproduzir”, os comandos.

Antes de gravar uma macro, planeje as etapas e os comandos que você deseja que a macro execute. Se cometer um erro durante a gravação da macro, as correções feitas também serão gravadas. Toda vez que você grava uma macro, ela é armazenada em um novo módulo anexado a uma pasta de trabalho.

Com o Editor do Visual Basic pode-se editar macros, copiar macros de um módulo para outro, copiar macros entre pastas de trabalho diferentes, renomear os módulos que armazenam as macros ou renomear as macros.

Gravar uma macro

1. No menu *Ferramentas*, aponte para *Macro* e, em seguida, clique em *Gravar nova macro*.
2. Na caixa *Nome da macro*, insira um nome para a macro. O primeiro caracter do nome da macro deve ser uma letra. Os demais caracteres podem ser letras, números ou caracteres sublinhados. Não são permitidos espaços no nome de uma macro; um caracter sublinhado funciona da mesma forma que um separador de palavras



3. Para executar a macro pressionando uma tecla de atalho do teclado, insira uma letra na caixa *Tecla de atalho*. Use CTRL + letra (para as letras minúsculas) ou CTRL + SHIFT + letra (para as letras maiúsculas). A letra da tecla de atalho não pode ser um número ou caracter especial. A tecla de atalho substituirá quaisquer teclas de atalho padrão do Microsoft Excel enquanto a pasta de trabalho que contém a macro estiver aberta.
4. Na caixa *Armazenar macro em*, clique no local onde você deseja gravar a macro. Se você deseja que uma macro fique disponível sempre que usar o Microsoft Excel, grave a macro na pasta de trabalho pessoal de macros da pasta XLInício. Para incluir uma descrição da macro, digite a descrição na caixa *Descrição*.
5. Clique em *OK*. Se você pressionar células durante a execução de uma macro, a macro selecionará as mesmas células independentemente das células que foram selecionadas pela primeira vez, pois ela grava as referências absolutas de célula. Se você desejar que uma macro selecione células independentemente da posição da célula ativa quando a macro estiver sendo executada, defina o gravador de macro para gravar as referências relativas de célula. Na barra de ferramentas *Parar gravação*, clique em *Referência relativa*. O Microsoft Excel continuará a gravar macros com referências relativas até que você saia do Microsoft Excel ou até que você clique em *Referência relativa* novamente.
6. Execute as ações que você deseja gravar.
7. Na barra de ferramentas *Parar gravação*, clique em *Parar gravação*.

Dica: Se desejar que uma macro selecione uma célula específica, execute uma ação e, em seguida, selecione outra célula relativa à célula ativa, pode-se combinar o uso de referências absolutas e relativas quando for gravar a macro. Basta iniciar a gravação com *Referência relativa* não pressionada e ativá-la tão logo terminem os comandos que se deseja realizar em modo absoluto.

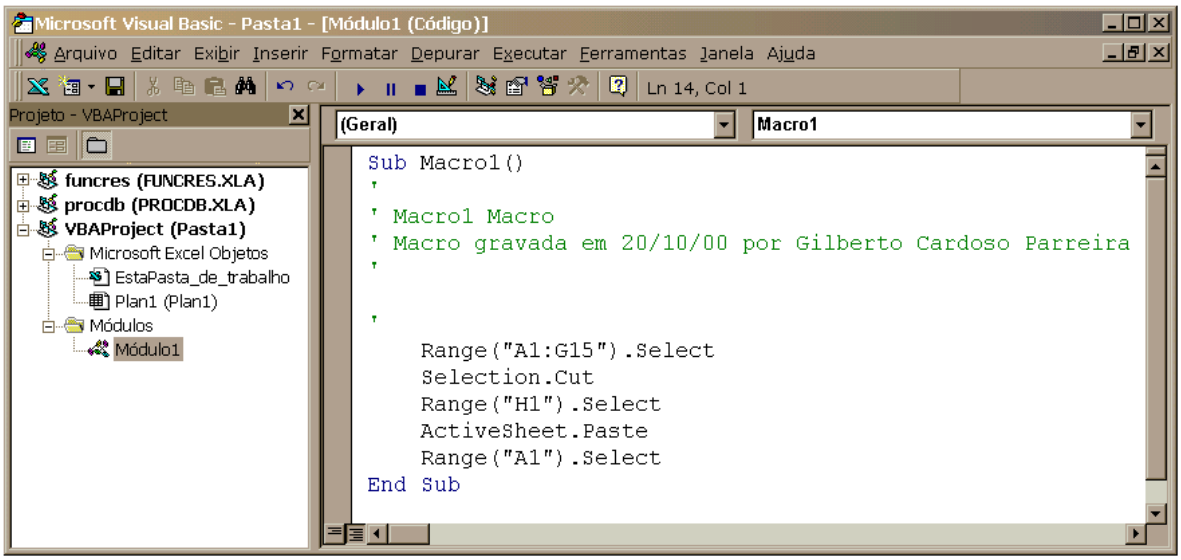
Editar uma macro

Antes de editar uma macro, familiarize-se com o Editor do Visual Basic. O Editor do Visual Basic pode ser usado para gravar e editar macros anexadas às pastas de trabalho do Microsoft Excel.

1. No menu *Ferramentas*, aponte para *Macro* e, em seguida, clique em *Macros*.
2. Na caixa *Nome da macro*, insira o nome da macro.
3. Clique em *Editar*.

A janela a seguir mostra o *Editor de Códigos do VBA*. Do lado esquerdo temos o "*Explorer do Projeto*" e do lado direito temos a "*Janela do Editor de Código*".

Se você não estiver vendo a janela como abaixo, clique em Exibir e "*Explorer do projeto*" ou CTRL-R e, em seguida, clique em *Exibir* e "*Código*"; ou F7.



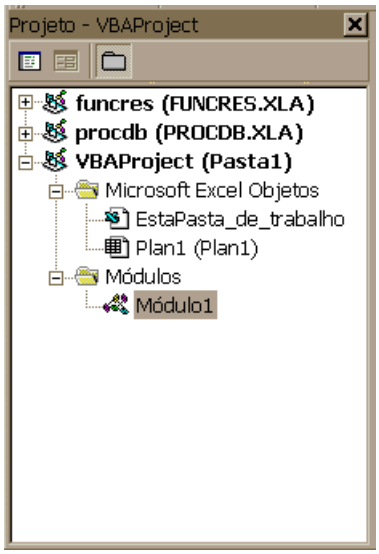
Explorer do projeto

Exibe uma lista hierárquica dos projetos e todos os seus itens contidos e mencionados por cada um deles.

Elementos da janela: os 3 botões na parte superior da janela do projeto são:

- *Visualizar código*: exibe a janela Código para que você edite e grave o código associado ao ítem selecionado.
- *Visualizar objeto*: exibe a janela Objeto do item selecionado, um documento existente ou o UserForm.
- *Alternar pastas*: Oculta e mostra as pastas de objeto enquanto continua a mostrar os itens individuais contidos neles.

Podemos listar todos os projetos carregados e os itens incluídos em cada um deles.



A janela de código

Utilize a janela *Código* para gravar, exibir e editar código do Visual Basic. Pode-se abrir o mesmo número de janelas *Código* como de módulos, de modo a poder visualizar facilmente o código em diferentes formulários ou módulos e copiar e colar entre eles.

Pode-se abrir uma janela *Código* a partir da:

- Janela *Projeto*, selecionando um formulário ou um módulo, e escolhendo o botão *Visualizar Código*.
- Janela *UserForm*, clicando duas vezes em um controle ou formulário, escolhendo *Código* no menu *Exibir* ou pressionando F7.

Pode-se arrastar o texto selecionado para:

- Um local diferente na janela *Código* atual.
- Outra janela código
- As janelas Imediata e Inspeção
- A lixeira

Elementos da janela

- *Caixa Objeto*: exibe o nome do objeto selecionado. Clique na seta à direita da caixa de listagem para exibir uma lista de todos os objetos associados ao formulário.
- *Caixa Procedimentos/Eventos*: Lista todos os eventos reconhecidos pelo Visual Basic de um formulário ou controle exibido na caixa *Objeto*. Quando você seleciona um evento, o procedimento de evento associado a esse nome de evento é exibido na janela *Código*. Caso (Geral) seja exibido na caixa *Objeto*, a caixa *Procedimento* listará as declarações e todos os procedimentos gerais que tenham sido criados para o formulário. Caso você esteja editando o código do módulo, a caixa *Procedimento* listará todos os procedimentos gerais no módulo. Em ambos os casos, o procedimento selecionado na caixa *Procedimento* é exibido na janela *Código*.

Todos os procedimentos em um módulo aparecem em uma lista de rolagem, classificada em ordem alfabética pelo nome. A seleção de um procedimento nas caixas de listagem suspensas na parte superior da janela Código move o cursos para a primeira linha de código no procedimento selecionado.

Barra de divisão

Quando esta barra é arrastada para baixo, divide a janela *Código* em dois painéis horizontais, que podem ser rolados de modo independente. Em seguida, você poderá visualizar diferentes partes do código ao mesmo tempo. As informações que aparecem na caixa Objeto e na caixa *Procedimentos/Eventos* se aplicam ao código no painel que tem o foco. Para fechar um painel, arraste a barra para a parte superior ou inferior da janela ou clique duas vezes sobre ela.

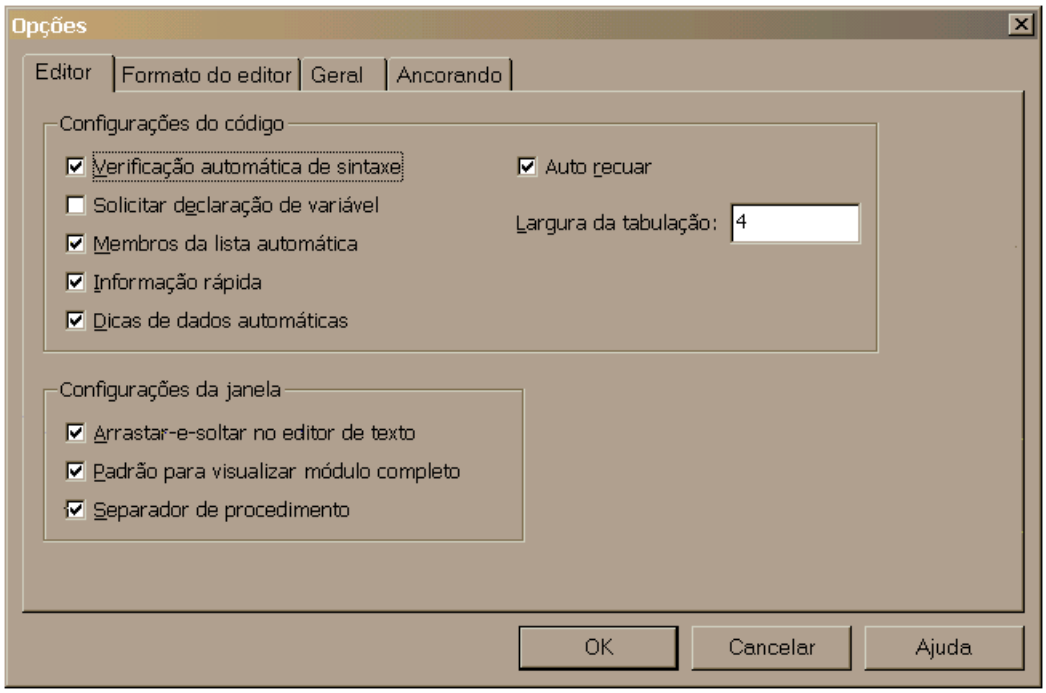
Barra do Indicador de margem

Uma área cinzenta no lado esquerdo da janela *Código*, onde são exibidos os indicadores de margem.

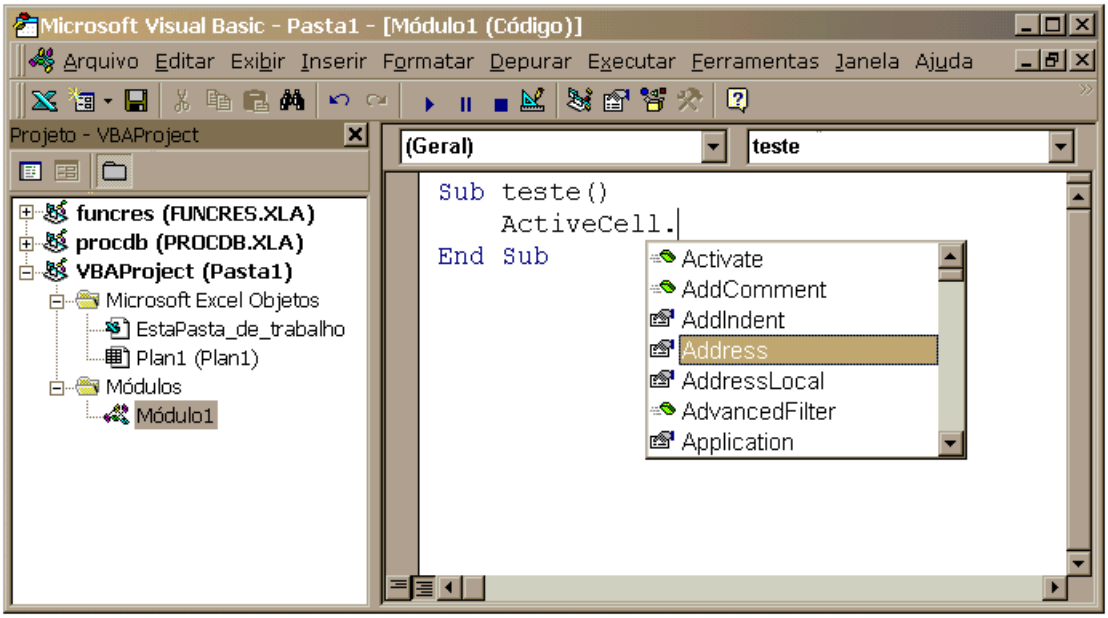
- Ícone *Visualizar procedimento*: exibo e procedimento selecionado. A janela *Código* só exibe um procedimento de cada vez.
- Ícone *Visualizar módulo completo*: exibe todo o código do módulo.

Digitando código

Para digitar código na janela do editor de códigos, pode-se usar as dicas automáticas. Normalmente elas estão ativadas, mas verifique dentro do editor do VBA, menu Ferramentas – Opções se o item "Informação rápida" está marcado.



Voltando à janela do Editor, quando preencher uma instrução válida, por exemplo:



Logo após digitar o ponto do comando "ActiveCell" surge a lista das opções válidas. Usando as setas pode-se selecionar o comando desejado e ao teclar "Espaço" a opção selecionada será apresentada. Por exemplo, selecione "Address". Logo em seguida as opções do subcomando

serão exibidas. Após a escolha certa, pressione novamente a barra de espaços e assim até terminar a linha de comando.

Executar uma macro

Para executar uma macro no Microsoft Excel:

- Abra a pasta de trabalho que contém a macro
- No menu *Ferramentas*, aponte para *Macro* e, em seguida, clique em *Macros*
- Na caixa **Nome da Macro**, insira o nome da macro que se deseja executar
- Clique em *Executar*

Observação: Para interromper uma macro antes que ela conclua suas ações, pressione ESC.

Adicionar controles a uma planilha

Pode-se adicionar controles a uma planilha, usando a barra de ferramentas *Formulários*. Esta barra contém controles elaborados para trabalhar no Microsoft Excel e com macros preexistentes. Use um controle da barra de ferramentas *Formulários* quando você precisar de um controle para executar uma única macro ou quando criar um site da Web.

A caixa de ferramentas de controle contém os controles ActiveX e os controles personalizados, instalados por outros aplicativos. Os controles ActiveX usam macros que você escreveu especificamente para o controle ActiveX. Use um controle ActiveX quando você precisar controlar diferentes eventos que ocorrem quando você usa o controle.

Ao adicionar controles, pode-se alterar as propriedades dos controles. As propriedades de um controle definem a sua aparência, a célula ou o intervalo de células ao qual o controle se refere, e ao estado do controle (por exemplo, se uma caixa de seleção estará marcada ou desmarcada por padrão).

A partir do menu *Exibir* clique em *Barra de Ferramentas* e ative a barra *Formulários*. Se preferir usar o atalho, clique com a tecla direita do mouse sobre qualquer barra de ferramentas visível e seleciona a barra *Formulários*.

Posicione a barra de formulários em um local conveniente da tela, evitando ancora-la nos extremos da janela (isto não será necessário, já que esta é uma barra de pouco uso).

Tipos de botões, caixas de seleção e outros controles na barra de Formulários

Os botões, caixas de seleção e outros controles disponíveis na barra de ferramentas Formulários oferecem opções quando você usa um formulário. Por exemplo, se você estiver criando um formulário de pessoal, poderá adicionar dois botões de opção para especificar se um empregado trabalha meio expediente ou em horário integral. Como só é possível selecionar um botão de opção de cada vez, o empregado pode trabalhar meio expediente ou em horário integral, mas não em ambos.

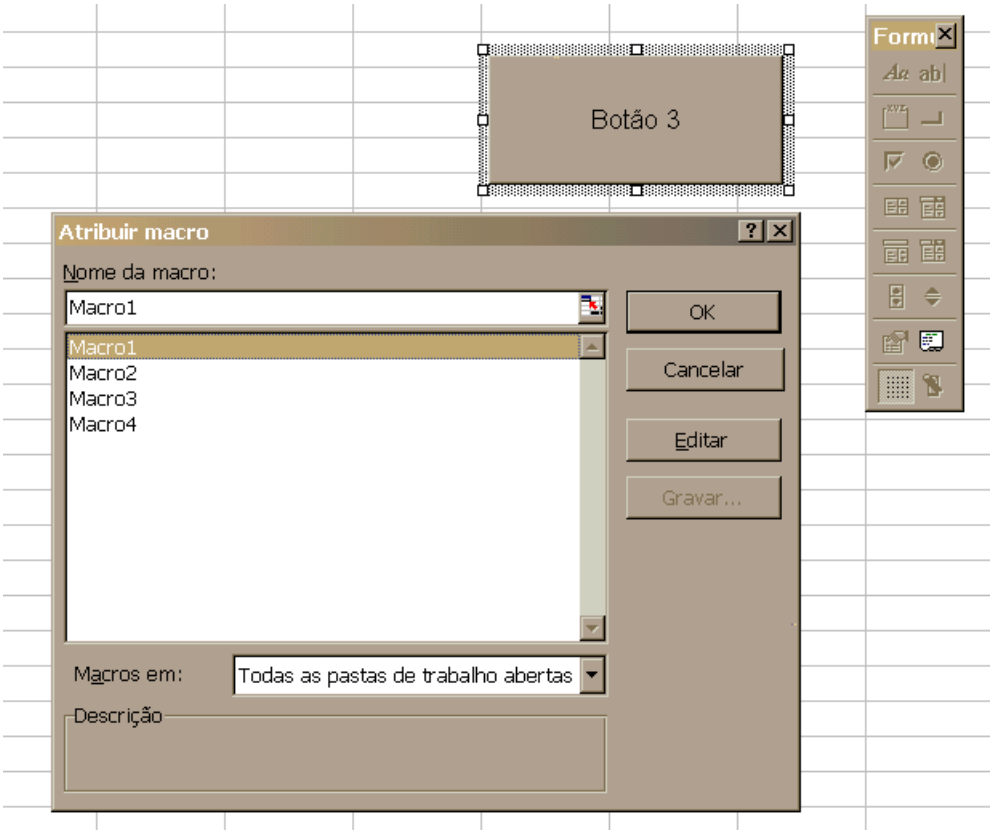
A maioria dos controles tem propriedades que pode-se alterar; no entanto, você só pode modificar as propriedades de formatação dos controles botão de comando e rótulo. A seguinte tabela lista os controles ActiveX que podem ser encontrados na *Caixa de ferramentas de controle*. Para obter maiores informações sobre os controles ActiveX, consulte a Ajuda do Visual Basic.

Representação	Tipo de controle	Descrição
	Caixa de seleção	Uma opção que pode-se ativar ou desativar, marcando ou demarcando-a. Pode-se Ter mais de uma caixa de seleção

		marcada em uma planilha por vez.
	Caixa de texto	Uma caixa em que pode-se digitar texto
	Botão de comando	Um botão que inicia uma ação quando pressionado
	Botão de opção	Um botão usado para selecionar apenas uma opção dentro de um grupo de alternativas.
	Caixa de listagem	Uma caixa que contêm uma lista de itens.
	Caixa de combinação	Uma caixa de texto com uma caixa de listagem drop-down. Pode-se digitar ou selecionar uma opção na caixa.
	Botão Alternar	Um botão que permanece pressionado quando clicado e depois é liberado quando clicado novamente.
	Botão de rotação	Um botão que pode ser anexado a uma célula ou a uma caixa de texto. Par aumentar um valor, clique na seta para cima; para diminuir um valor, clique na seta para baixo
	Barra de rolagem	Um controle que percorre uma lista de valores quando se clica nas setas de rolagem ou quando se arrasta a caixa de rolagem. Pode-se percorreu uma página de valores, clicando entre a caixa de rolagem e uma das setas de rolagem
	Rótulo	Um texto adicionado a uma planilha ou a um formulário, que fornece informações sobre um controle ou sobre a planilha ou o formulário
	Imagem	Um controle que incorpora uma figura em um formulário
	Moldura	Uma borda e um rótulo que agrupam controles relacionados, como botões de opção ou caixas de seleção.

Para nosso exemplo, deve-se inserir um botão de comando. Selecione-o na barra de ferramentas e em seguida, desenhe um retângulo dentro da planilha, no canto superior direito.

Imediatamente ao soltar o mouse temos a seguinte janela:



Nesta janela podemos selecionar da lista de macros existentes aquela que será aplicada ao botão. Ou podemos criar ma nova macro a partir da tecla Novo ou ainda deixar esta atribuição para mais tarde clicando no botão *Cancelar*. Selecione uma das macros que você tiver criado. Poderemos mais tarde alterar a opção.

Usando referências relativas e referências absolutas

Uma referência identifica uma célula ou um intervalo de células em uma planilha e informa ao Microsoft Excel onde procurar valores ou dados que você deseja usar em uma fórmula. Com as referências, pode-se usar os dados contidos em outras partes de uma planilha em uma fórmula ou usar o valor de uma célula em diversas fórmulas. Pode-se também fazer referências a células em outras planilhas da mesma pasta de trabalho, a outras pastas de trabalho e a dados em outros programas. As referências às células em outras pastas de trabalho são denominadas referências externas. As referências aos dados em outros programas são denominadas referências remotas.

Por padrão, o Microsoft Excel usa o estilo de referência A1, que rotula colunas com letras (A a IV, para um total de 256 colunas) e rotula linhas com números (1 a 65536). Para fazer referência a uma célula, insira a letra da coluna seguida do número da linha. Por exemplo, D50 refere-se à célula na interseção da coluna D e linha 50. Para fazer referência a um intervalo de células, insira a referência da célula no canto superior esquerdo do intervalo, dois pontos (:) e depois a referência da célula no canto inferior direito do intervalo. A seguir estão exemplos de referências:

Para fazer referência	Use
À célula na coluna A e linha 10	A10
Ao intervalo de células na coluna A e linhas 10 a 20	A10:A20
Ao intervalo de células na linha 15 e colunas B a E	B15:E15

A todas as células na linha 5	5:5
A todas as células nas linhas 5 a 10	5:10
A todas as células na coluna H	H:H
A todas as células nas colunas H a J	H:J

Você também pode usar um estilo de referência onde tanto as linhas quanto as colunas da planilha são numeradas. O estilo L1C1 é útil para computar posições de linha e coluna em macros e pode ser útil para mostrar as referências relativas da célula. No estilo L1C1, o Microsoft Excel indica a posição de uma célula com um "L" seguido de um número de linha e um "C" seguido de um número de coluna.

Observações: Dependendo da tarefa que você deseja executar no Microsoft Excel, use as referências relativas da célula, que são referências a células relativas à posição da fórmula, ou referências absolutas, que são referências de célula que sempre referem-se a células em uma localização específica.

Use os rótulos de colunas e linhas em uma planilha para fazer referências a células dentro destas colunas e linhas, ou crie nomes descritivos para representar células, intervalos de células, fórmulas ou valores constantes.

Se você desejar analisar dados na mesma célula ou intervalo de células em diversas planilhas da pasta de trabalho, use uma referência 3-D. Uma referência 3-D inclui a referência de célula ou intervalo, precedida de um intervalo de nomes de planilha. O Microsoft Excel usa quaisquer planilhas armazenadas entre o nome inicial e o nome final da referência.

Referências absolutas e relativas em código VBA

Os dois códigos a seguir mostram a diferença implementada pelo Excel quando optamos por uma ou outra opção:

```
Sub ReferenciaRelativa()
    ActiveCell.Offset (-10, -5).Range("A1:B2").Select
    Selection.Cut Destination:=ActiveCell.Offset(3, 0).Range("A1:B2")
    ActiveCell.Offset(3, 0).Range("A1:B2").Select
End Sub
O próximo código foi gravado com o botão de referência relativa
desativado:
Sub ReferenciaAbsoluta()
    Range("A1:B2").Select
    Selection.Cut Destination:= Range("A4:B5")
    Range("A4:B5").Select
End Sub
```

Compreendendo a sintaxe do Visual Basic

A sintaxe em um tópico da Ajuda do Visual Basic de um método, instrução ou função mostra todos os elementos necessários para que o método, instrução ou função seja utilizada corretamente. Os exemplos deste tópico ensinam a interpretar os elementos de sintaxe mais comuns.

Sintaxe do comando Activate

objeto.Activate

Na sintaxe do método Activate, a palavra destacada em itálico (objeto) é um marcador das informações que você fornece; neste caso, um código que retorna um objeto. As palavras destacadas em negrito devem ser digitadas tal e qual aparecem. Por exemplo, o procedimento a seguir ativa a segunda janela no documento ativo.

```
Sub TornarAtiva()
    Sheets(2).Activate
End Sub
```

Sintaxe da função MsgBox

MsgBox (prompt[,buttons] [,title] [.helpfile, context])

Na sintaxe da função MsgBox as palavras destacadas em negrito/itálico são os argumentos nomeados da função. Os argumentos colocados entre colchetes são opcionais (não digite os colchetes no código Visual Basic). O único argumento que você deve fornecer para a função MsgBox é o texto do aviso.

Os argumentos das funções e métodos podem ser especificados no código pela posição ou pelo nome. Para especificar os argumentos por posição, siga a ordem apresentada na sintaxe, separando cada argumento com uma vírgula, como no exemplo apresentado a seguir:

```
MsgBox "A sua resposta está correta !",0,"Caixa de resposta"
```

Para especicar um argumento pelo nome utilize o nome do argumento seguido por dois pontos (:) e um sinal de igualdade (=) e o valor do argumento. Pode-se especificar os argumentos nomeados em qualquer ordem, como no exemplo apresentado a seguir:

```
MsgBox Title:="Caixa de Resposta", Prompt:="A sua resposta está
correta !"
```

A sintaxe das funções e de alguns métodos mostra os argumentos colocados entre parênteses. Como essas funções e métodos retornam valores, você deve colocar os argumentos entre parênteses para atribuir o valor a uma variável. Caso você ignore o valor de retorno ou não passe nenhum tipo de argumento, não inclua os parênteses. Os métodos que não retornam valores não precisam que os seus argumentos sejam colocados entre parênteses. Essas orientações se aplicam quer você esteja utilizando argumentos posicionais ou nomeados.

No exemplo a seguir, o valor de retorno da função MsgBox é um número que indica que o botão selecionado está armazenado na variável minhaVar. Como o valor de retorno é utilizado, os parênteses são obrigatórios. Em seguida, outra caixa de mensagem exibirá o valor da variável.

```
Sub Pergunta()
    minhaVar = MsgBox(Prompt:="Gosto do meu emprego.", _
        Title:="Caixa de Resposta ", Botões:="4")
    MsgBox minhaVar
End Sub
```

Sintaxe da instrução Option Compare

Option Compare {Binary | Text | Database }

Na sintaxe da intrução Option Compare, as chaves e a barra vertical indicam uma escolha obrigatória entre três itens (não digite as chaves na instrução do Visual Basic). Por exemplo, a

instrução apresentada a seguir especifica que, dentro do módulo, as seqüências de caracteres são comparadas em uma ordem de classificação que não faz distinção entre letras maiúsculas ou minúsculas.

```
Option Compare Texto
```

Sintaxe da instrução Dim

Dim nomedavariável [([subscritos])] [As tipo] [,nomedavariável [([subscritos])] [As tipo]] ...

Na sintaxe da instrução Dim, a palavra Dim é uma palavra-chave obrigatória. O único elemento obrigatório é nomedavariável (o nome da variável). Por exemplo, a instrução apresentada a seguir cria três variáveis: minhaVar, proximaVar e terceiraVar. Elas são automaticamente declaradas como variáveis do tipo Variant.

```
Dim minhaVar, proximaVar, terceiraVar
```

O exemplo apresentado a seguir declara uma variável como uma String. A inclusão de um tipo de dados poupa a memória e pode ajudá-lo a encontrar erros no seu código.

```
Dim minhaResposta as String
```

Para declarar diversas variáveis em uma instrução, insira o tipo de dados para cada variável. As variáveis declaradas sem um tipo de dados são automaticamente declaradas como Variant.

```
Dim x as Integer, y as Integer, z as Integer
```

Na instrução apresentada a seguir, x e y são atribuídos ao tipo de dados Variant. Somente z é atribuido ao tipo de dados Integer.

```
Dim x, y, z as Integer
```

Caso você esteja declarando uma variável de matriz, você deve colocar parênteses. Os subscritos são opcionais. A instrução apresentada a seguir dimensiona uma matriz dinâmica, minhaMatriz.

```
Dim minhaMatriz ()
```

A linguagem do VBA

O gravador de macros é uma excelente ferramenta para criar macros. Mas existem algumas limitações de comandos que não podemos gravar automaticamente, tais como:

- Ramificações condicionais usando IF..THEN
- Estruturas de loop tais como FOR..NEXT
- Referências e cálculos complexos
- Funções e caixas de diálogo, etc.

Para realizar tais operações precisamos inserir código manualmente. E para tal, devemos aprender alguns conceitos fundamentais sobre objetos, propriedades e métodos.

Objetos, propriedades, métodos e eventos

Um objeto representa um elemento de um aplicativo, como uma planilha, uma célula, um gráfico, um formulário ou um relatório. No código do Visual Basic você deve identificar um objeto antes de poder aplicar um dos métodos do objeto ou alterar o valor de uma das suas propriedades.

Uma coleção é um objeto que contém uma série de outros objetos, geralmente (mas não sempre) do mesmo tipo. No Microsoft Excel, por exemplo, o objeto *Workbooks* contém todos os objetos *Workbook* que estiverem abertos. No Visual Basic a coleção *Forms* contém todos os objetos *Form* de um aplicativo.

Os itens de uma coleção podem ser identificados por números ou por nome. Por exemplo, no procedimento apresentado a seguir, *Workbooks*(1) identifica o primeiro objeto *Workbook* aberto.

```
Sub FecharPrimeiro()
    Workbooks(1).Close
End Sub
```

O procedimento apresentado a seguir utiliza um nome especificado como uma seqüência de caracteres para identificar um objeto *Form*

```
Sub FecharFormulario()
    Forms("MeuFormulario.frm").Close
End Sub
```

Pode-se manipular toda uma coleção de objetos caso os objetos compartilhem métodos. Por exemplo, o procedimento apresentado a seguir fechar todos os formulários abertos.

```
Sub FecharTodos()
    Forms.Close
End Sub
```

Um método é uma ação que um objeto pode executar. Por exemplo, *Add* é um método do objeto *ComboBox*, pois posiciona uma nova entrada a um quadro de combinação.

O procedimento apresentado a seguir utiliza o método *Add* para adicionar um novo item a *ComboBox*.

```
Sub AdicionarEntrada(newEntrada as String)
    Combo1.Add newEntrada
End Sub
```

Uma propriedade é um atributo de um objeto que define uma de suas características (assim como tamanho, cor ou localização na tela) ou um aspecto de seu comportamento (assim como se ele está ativo ou visível). Para alterar as características de um objeto, pode-se alterar os valores de suas propriedades.

Para definir o valor de uma propriedade, coloque, após a referência a um objeto, um ponto (.), o nome da propriedade que deseja definir, um sinal de igualdade (=) e o novo valor da propriedade. Por exemplo, o procedimento a seguir altera o texto de título de um formulário do Visual Basic através da definição da propriedade *Caption*:

```
Sub AlterarNome (newTitulo)
    Meuformulario.Caption  = newTitulo
End Sub
```

Algumas das propriedades não podem ser definidas. O tópico da Ajuda de cada propriedade indica se ela pode ser definida (leitura e gravação), se pode somente ser lida (somente leitura) ou se pode somente ser gravada (somente gravação).

Para recuperar informações sobre um objeto, bastar retornar o valor de uma de suas propriedades. O procedimento apresentado a seguir utiliza uma caixa de mensagem para exibir o título na parte superior do formulário ativo no momento.

```
Sub ObterNomedoFormulario ()
    NomedoFormulario = Screen.ActiveForm.Caption
    MsgBox NomedoFormulario
End Sub
```

Um evento é uma ação reconhecida por um objeto, como clicar o mouse ou pressionar uma tecla, para o qual pode-se gravar código para responder. Podem ocorrer eventos em resposta a uma ação do usuário ou a um código do programa. Eles também podem ser disparados pelo sistema.

Objetos do VBA

Entre os principais objetos do Excel VBA temos:

Objeto Application

Representa o aplicativo que está sendo executado no momento (no nosso caso ele representa o próprio EXCEL). O objeto *Application* contém:

1. Definiçoes e opções para o aplicativo como um todo (muitas das opções da caixa de diálogo *Opções*, do menu *Ferramentas*, por exemplo)
2. Métodos que retornem objetos do nível mais alto, como *ActiveCell*, *ActiveShhet* e assim por diante.

Como usar o objeto Application

Use a propriedade *Application* para retornar o objeto *Application*. O exemplo seguinte aplica a propriedade Windows ao objeto *Application*:

```
Application.Windows("Pasta1.xls").Activate
```

O exemplo seguinte cria um objeto de planilha do Microsoft Excel em um outro aplicativo e, em seguida, abre a pasta de trabalho do Microsoft Excel

```
Set xl = CreateObject ("Excel.Sheet")
xl.Application.Workbooks.Open "NovaPasta.xls"
```

Comentários: Muitas das propriedades e métodos que retornam os objetos mais comuns da interface do usuário, como a célula ativa (propriedade *ActiveCell*), podem ser usados sem o qualificador de objeto *Application*. Por exemplo, em vez de escrever

Application.ActiveCell.Font.Bold = True, pode-se escrever simplesmente ActiveCell.Font.Bold = True.

Objeto Worksheets

É o segundo maior objeto dentro do objeto *Application*. Uma coleção de todos os objetos *Worksheet* na pasta de trabalho ativa ou especificada. Cada objeto *Worksheet* representa uma planilha. Use a propriedade *Worksheets* para retornar a coleção *Worksheets*. O exemplo seguinte move todas as planilhas para o final da pasta de trabalho.

```
Worksheets.Move after:=Sheets(Sheet.Count)
```

Use o método *Add* para criar uma nova planilha e adicioná-la a coleção. O exemplo seguinte adiciona duas novas planilhas antes da planilha um da pasta de trabalho ativa.

```
Worksheets.Add count:=2, before:=Sheets(1)
```

Use *Worksheets*(*índice*) para retornar um único objeto *Worksheet*. O exemplo seguinte oculta a planilha um na pasta de trabalho ativa:

```
Worksheets(1).Visible = False
```

O objeto *Worksheet* é também um membro da coleção *Sheets*. A coleção *Sheets* contém todas as planilhas da pasta de trabalho (tanto folhas de gráfico quanto planilhas de trabalho).

As seguintes propriedades para retornar um objeto *Worksheet* são descritas nesta seção:

- Propriedade *Worksheets*
- Propriedade *ActiveSheet*

Propriedade Worksheets

Use *Worksheets(índice)* para retornar um único objeto *Worksheet*. O exemplo seguinte oculta a planilha um na pasta de trabalho ativa:

```
Worksheets(1).Visible = False
```

O número de índice da planilha denota a posição de uma planilha na barra de guias da pasta de trabalho. *Worksheets(1)* é a primeira planilha (mais a esquerda) e *Worksheets(Worksheets.Count)* é a última. Todas as planilhas são incluídas na contagem do índice, mesmo quando estão ocultas.

O nome da planilha é mostrado na guia da planilha. Use a propriedade *Name* para definir ou retornar o nome da planilha. O objeto *Worksheet* é também um membro da coleção *Sheets*. A coleção *Sheets* contém todas as planilhas da pasta de trabalho (tanto folhas de gráfico quanto planilhas de trabalho).

Propriedade ActiveSheet

Quando uma planilha é a planilha ativa, pode-se usar a propriedade *ActiveSheet* para referir-se a ela. O exemplo seguinte usa o método *Activate* para ativar Plan1, define a orientação da página como modo paisagem e, em seguida, imprime a planilha.

```
Worksheets("Plan1").Activate
ActiveSheet.PageSetup.Orientation = xlLandscape
ActiveSheet.PrintOut
```

Objeto Range

Representa uma célula, uma linha, uma coluna, uma seleção de células contendo um ou mais blocos contíguos de células ou um intervalo 3-D. As seguintes propriedades e métodos para retornar um objeto *Range* são descritas nesta seção:

- Propriedade *Range*
- Propriedade *Cells*
- *Range* e *Cells*
- Propriedade *Offset*
- Método *Union*

Propriedade Range

Use *Range(argumento),* onde argumento nomeia o intervalo, para retornar um objeto *Range* representando uma única célula ou um intervalo de células. O exemplo seguinte coloca o valor da célula A1 na célula A5.

```
Worksheets("Plan1").Range("A5").Value =
Worksheets("Plan1").Range("A1").Value
```

O exemplo seguinte preenche o intervalo A1:H8 com números randômicos definindo a fórmula para cada célula do intervalo. Quando usada sem um qualificador de objeto (um objeto a esquerda do ponto), a propriedade Range retorna um intervalo da planilha ativa. Se a planilha ativa não for uma planilha de trabalho, o método falhará. Use o método *Activate* para ativar uma planilha antes de usar a propriedade Range sem um qualificador de objeto explícito.

```
Worksheets("Plan1").Activate
Range("A1:H8").Formula = "=rand()"  ' O intervalo esta na planilha
ativa
```

O seguinte exemplo limpa o conteúdo do intervalo chamado "Criterios"

```
Worksheets(1).Range("Criterios").ClearContents
```

Se você usar um argumento de texto para o endereço do intervalo, você terá que especificar o endereço em notação de estilo A1 (você não poderá usar a notação de estilo L1C1.

Propriedade Cells

Use *Cells(linha,coluna)* para retornar uma única célula. O exemplo a seguir define o valor da célula A1 como 32:

```
Worksheets(1).Cells(1,1).Value = 32
```

O exemplo seguinte define uma fórmula para a célula A2:

```
Worksheets(1).Cells(2,1).Formula = "sum(b1:b5)"
```

Embora você também possa usar *Range ("A1")* para retornar a célula A1, pode haver ocasiões em que a propriedade *Cells* seja mais conveniente porque pode-se usar uma variável para a linha ou coluna. O exemplo seguinte cria cabeçalhos de coluna e linha na Plan1. Observe que após a planilha ser ativada, a propriedade *Cells* pode ser usada sem uma declaração explícita de planilha (ela retorna uma célula na planilha ativa)

```
Sub ConfigTabela()
Worksheets("Plan1").Activate
For oAno = 1 to 5
    Cells(1, oAno + 1).Value = 1990 + oAno
Next oAno

For oTrimestre = 1 to 4
    Cells(oTrimestre+1, 1).Value = "T" & oTrimestre
Next oTrimestre
End Sub
```

Apesar de você poder usar funções de cadeia de caracteres do Visual Basic para alterar as referências de estilo A1, é muito mais fácil (e é uma prática de programação muito melhor) usar a notação *Cells* (1,1)

Use *expressáo.Cells(linha,coluna)* onde expressão é uma expressão que retorne um objeto *Range*, para retornar parte de um intervalo. O exemplo seguinte define a fórmula para a célula C5:

```
Worksheets(1).Range("c5:c10").Cells(1,1).Formula = "=rand()"
```

Range e Cells

Use *Range(célula1, célula2)* onde *célula1* e *célula2* são objetos *Range* que especificam as células inicial e final para retornar um objeto *Range*. O exemplo seguinte define o estilo da linha da borda das células A1:J10:

```
With Worksheets(1)
    .Range(.Cells(1,1), .Cells(10,10)).Borders.LineStyle = xlThick
End With
```

Observe o ponto na frente de cada ocorrência da propriedade *Cells*. O ponto será obrigatório se o resultado da instrução *With* anterior for aplicado à propriedade *Cells* nesse caso, para indicar que as células estão na planilha um (sem o ponto, a propriedade *Cells* retornaria as células da planilha ativa).

Propriedade Offset

Use *Offset(linha, coluna)* onde linha e coluna são os deslocamentos de linha e coluna, para retornar um intervalo em um deslocamento especificado em um outro intervalo. O exemplo seguinte seleciona a célula três linhas abaixo e uma coluna a esquerda da célula do canto superior esquerdo da seleção atual. Você não pode selecionar uma célula que não esteja na planilha ativa, portanto você precisa ativar primeiro a planilha.

```
Worksheets("Plan1").Activate
Selecion.Offset(3,1).Range("A1"),Select
```

Método Union

Use *Union(intervalo1, intervalo2, ...)* para retornar intervalos de várias áreas, isto é, intervalos compostos de dois ou mais blocos contíguos de células. O exemplo seguinte cria um objeto definido como a união de intervalos A1:B2 e C3:D4 e, em seguida, seleciona o intervalo definido

```
Dim i1 as Range, i2 as Range, meuIntervalo as Range
Worksheets("Plan1").Activate
Set i1 = Range("A1:B2")
Set i1 = Range("C3:D4")
Set meuIntervalo  = Union(i1, i2)
MeuIntervalo.Select
```

Se você trabalha com seleções que contém mais do que uma área, a propriedade *Areas* é muito útil. Ela divide uma seleção de várias áreas em objetos Range individuais e, em seguida, retorna os objetos como uma coleção. Pode-se usar a propriedade *Count* na coleção retornada para verificar uma seleção que contenha mais do que uma área, como mostrado no exemplo seguinte:

```
Sub NenhumaSelDeVariasAreas()
    numeroDeAreasSelecionadas = Selection.Areas.Count
    If numeroDeAreasSelecionadas > 1 Then
        MsgBox "Você não pode executar este comando" & _
            "em seleções de várias áreas"
    End If
End Sub
```

Os cientistas nos dizem que...

Uma pessoa gasta, em média:

- 9.5 anos dormindo
- 4.2 anos comendo
- 3.8 anos no banheiro
- 2.8 anos viajando

e...

- 1.9 anos esperando o Excel recalcular!

A linguagem VBA

A linguagem do VBA corresponde a linguagem do Visual Basic e do Access Basic. Desta forma, um programa escrito em qualquer um destes sistemas não terá nenhuma barreira ao ser executado ou mesmo importado para dentro o Excel e aí executado.

Toda linguagem possui um tratamento de dados armazenados em memória. Estes dados são as variáveis do sistema. Estas variáveis devem ser declaradas previamente, para que possam ser usadas dentro do programa VBA.

Ao declarar variáveis, você geralmente irá utilizar uma instrução *Dim*. Estas restrições de declaração podem ser colocadas em um procedimento para criar uma variável em nível de procedimento, ou na parte superior de um módulo, na seção Declarações, para criar uma variável a nível de módulo.

O exemplo apresentado a seguir cria a variável *strNome* e especifica o tipo de dados *String*

```
Dim strNome as String
```

Caso esta instrução esteja dentro de um procedimento, a variável *strNome* poderá ser utilizada somente nesse procedimento. Se a instrução for colocada na seção Declarações do módulo, a variável *strNome* estará disponível para todos os procedimentos dentro do módulo, mas não para outros módulos no projeto. Para tornar essa variável disponível para todos os procedimentos no projeto, preceda-o com a instrução *Public*, como no exemplo apresentado a seguir:

```
Public strNome as String
```

Para obter maiores informações sobre a nomeação das suas variáveis, consulte regras de nomeação do Visual Basic, na ajuda do Visual Basic.

As variáveis podem ser declaradas como um dos seguintes tipos de dados: *Boolean*, *Byte*, *Integer*, *Long*, *Currency*, *Single*, *Double*, *Date*, *String* (para seqüências de caracteres de comprimento variável, String * comprimento (para seqüências de caracteres de comprimento fixo, *Object* ou *Variant*. Caso você não especifique um tipo de dados, o tipo *Variant* será atribuído como padrão. Você também poderá criar um tipo definido pelo usuário através da instrução *Type*. Para obter maiores informações sobre tipos de dados, consulte “Resumo de tipos de dados” na Ajuda do Visual Basic.

É possível declarar diversas variáveis em uma instrução. Para especificar um tipo de dados, você precisa incluir o tipo de dados para cada variável. Na instrução apresentada a seguir, as variáveis *intX*, *intY* e *intZ* são declaradas como *Integer*:

```
Dim intX as Integer, intY as Integer, intZ as Integer
```

Na instrução apresentada a seguir, *intX* e *intY* são declaradas como tipo *Variant*; apenas *intZ* é declarada como tipo *Integer*.

```
Dim intX, intY, intZ as Integer
```

Você não precisa fornecer o tipo de dados da variável na instrução de declaração. Caso você omita o tipo de dados, a variável será do tipo *Variant*.

Utilizando a instrução Public

Você não pode utilizar a instrução *Public* para declarar as variáveis públicas em um nível de módulo.

```
Public strName as String
```

As variáveis públicas podem ser utilizadas em todos os procedimentos do projeto. Caso uma variável pública seja declarada em um módulo padrão ou um módulo de classe, ela também poderá ser utilizada nos projetos que façam referência ao projeto onde a variável pública é declarada.

Utilizando a instrução Private

Pode-se utilizar a instrução *Private* para declarar as variáveis privadas em nível de módulo.

```
Private MeuNome as String
```

As variáveis privadas podem ser utilizadas somente por procedimentos do mesmo módulo.

Observação: Ao ser utilizada em nível de módulo, a instrução *Dim* é equivalente à instrução *Private*. Pode ser que você precise utilizar a instrução *Private* para facilitar a leitura e a interpretação do seu código.

Utilizando a instrução Static

Quando você utiliza a instrução *Static* no lugar de uma instrução *Dim*, a variável declarada reterá os seus valores entre as chamadas.

Utilizando a instrução Option Explicit

Para declarar implicitamente uma variável no Visual Basic, basta utilizá-la em uma instrução de atribuição. Todas as variáveis declaradas implicitamente são do tipo *Variant*. As variáveis do tipo *Variant* exigem mais recursos de memória que a maioria das outras variáveis. O seu aplicativo será mais eficiente se você declarar as variáveis explicitamente e com um tipo de dados especifico. A declaração explícita de todas as variáveis reduz a incidência de erros de nomeação conflitante e de digitação.

Para impedir que o Visual Basic faça declarações implícitas, você poderá inserir a instrução *Option Explicit* em um módulo antes de todos os procedimentos. Essa instrução obriga-o a declarar explicitamente todas as variáveis dentro do módulo. Caso um módulo contenha a instrução *Option Explicit*, ocorrerá um erro de tempo de compilação quando o Visual Basic se deparar com um nome de variável que ainda não tenha sido declarada ou que apresente algum erro de digitação.

Pode-se definir uma opção no ambiente de programação do Visual Basic para inserir automaticamente a instrução *Option Explicit* em todos os módulos novos. Para obter ajuda sobre as mudanças nas opções do ambiente Visual Basic, consulte a documentação do aplicativo. Observe que esta opção não altera o código existente que você já tenha escrito.

Tipos de dados

A tabela a seguir mostra os tipos de dados suportados, incluindo tamanhos e intervalos de armazenamento.

Byte	1 byte	0 a 255

Boolean	2 bytes	True ou False
Integer	2 bytes	-32768 a 32767
Long (inteiro longo)	4 bytes	-2147483648 a 2147483647
Single	4 bytes	-3,402823E38 a –1,401298E-45 para valores negativos 1,401298E-45 a 3,402823E38 para valores positivos
Double	8 bytes	-1,797E308 a –4,9406E-324 para valores negativos 4,9406E-324 a 1,797E308 para valores positivos
Currency	8 bytes	-922337203685477,5808 a 922337203685477,5807
Decimal	14 bytes	+/- 79.228.162.514.264.337.593.543.950.335 sem ponto decimal +/- 7,9228162514264337593543950335 com 28 casas decimais a direita
Date	8 bytes	De 01/01/100 a 31/12/9999
Object	4 bytes	Qualquer referência a object
String	10 bytes + comprimento da seqüência de caracteres	De 0 até aproximadamente 2 bilhões
Definido pelo usuário		

Observação: Matrizes de qualquer tipo de dados requerem 20 bytes de memória, mais 4 bytes para cada dimensão da matriz, mais o número de bytes ocupados pelos próprios dados. A memória ocupada pelos dados pode ser calculada multiplicando-se o número de elementos de dados pelo tamanho de cada elemento. Por exemplo, os dados em uma matriz de dimensão única consistindo de 4 elementos de dados *Integer* de 2 bytes cada ocupa 8 bytes. Os 8 bytes exigidos para os dados mais os 24 bytes fixos fazem com que o requisito de memória para a matriz seja de 32 bytes.

A menos que seja especificado de forma diferente as variáveis não declaradas são atribuídas ao tipo de dados *Variant*. Este tipo de dados facilita a gravação de programas, mas nem sempre é o tipo de dados de utilização mais eficiente.

Você deve levar em consideração a utilização de outros tipos de dados se:

O seu programa for muito grande e utilizar muitas variáveis.

O seu programa tiver que ser executado com o máximo de velocidade possível.

Você gravar dados diretamente em arquivos de acesso aleatório.

Além de *Variant*, são estes os tipos de dados suportados: Byte, *Boolean*, *Integer*, *Long*, *Single*, *Double*, *Currency*, *Decimal*, *Date*, *Object* e *String*. Utilize a instrução *Dim* para declarar uma variável de um tipo específico, como no exemplo apresentado a seguir:

```
Dim X as Integer
```

Essa instrução declara que uma variável X é um inteiro – um número inteiro entre –32768 e 32767. Caso você tente definir X como um número fora desse intervalo, ocorrerá um erro. Caso você tente definir X como uma fração, o número será arredondado. Por exemplo:

```
X = 32768    ‘ Provoca erro
X = 5,9      ‘ Define X como 6
```

Programação estruturada

A linguagem do VBA permite a construção de subprocedimentos contendo toda a estrutura de uma linguagem eficiente e maleável ao mesmo tempo. Para tal ela possui os comandos padrão das demais linguagens como Pascal, Basic, Clipper, Delphi, etc. As estruturas são:

- IF ... THEN ... ELSE
- SELECT CASE ... CASE ... ENDCASE
- DO ... LOOP
- FOR ... TO ... NEXT
- FOR ... EACH ... NEXT
- WITH ... END WITH

Vamos detalhar cada um deles a seguir:

If ... Then ... Else

Executa condicionalmente um grupo de instruções, dependendo do valor de uma expressão

Sintaxe

```
If condição Then [Instruções] [Else Instruçõeselse]
```

Ou pode-se utilizar a sintaxe de formato de bloco:

```
If condição Then
[Instruções]
[ElseIf condição-n Then
[InstruçõesElseIf] ...
[Else
[InstruçõesElse]]
End If
```

A instrução *If ... Then ... Else* possui as partes a seguir:

Parte	Descrição
Condição	Obrigatória. Um ou mais dos tipos de expressão seguintes: Uma expressão numérica ou expressão de sequência de caracteres que é avaliada como True ou False. Se condição for Null, será tratada como False. Uma expressão do formato TypeOf NomedoObjeto Is TipodoObjeto. O NomedoObjeto é qualquer referência a objeto e TipodoObjeto é qualquer tipo de objeto válido. A expressão será True se NomedoObjeto for o tipo de objeto especificado por TipodoObjeto;

	caso contrário, será False
Instruções	Opcional em formato de bloco; requerida em formato de uma só linha que não possui cláusula *Else*. Uma ou mais instruções separadas por dois-pontos; executada se condição for *True*.
Condição-n	Opcional. Igual a condição
InstruçãoElseIf	Opcional. Uma ou mais instruções executadas se a condição-n associada for True.
InstruçãoElse	Opcional. Uma ou mais instruções executadas se nenhuma expressão condição ou condição-n anterior for *True*.

Comentários: Pode-se utilizar o formato de uma só linha (primeira sintaxe) para testes simples e curtos. Entretanto, o formato de bloco (segunda sintaxe) proporciona mais estrutura e flexibilidade do que o formato de uma só linha e normalmente é mais fácil de ler, manter e depurar.

Com o formato de uma só linha é possível ter múltiplas instruções executadas como resultado de uma decisão *If...Then*. Todas as instruções devem estar na mesma linha e separadas por dois-pontos, como na seguinte instrução:

```
If A > 10 Then A = A + 1 : B = B + A : C = C + B
```

Uma instrução *If* em formato de bloco deve ser a primeira em uma linha. As partes *Else*, *ElseIf* e *End If* da instrução podem ter somente um número de linha ou rótulo de linha precedendo-as. O *If* em bloco deve terminar com uma instrução *End If*.

Para determinar se uma instrução é ou não *If* em bloco, examine o que vem em seguida à palavra-chave *Then*. Se qualquer coisa exceto um comentário aparecer depois de *Then* na mesma linha, a instrução será tratada como instrução *If* de uma só linha.

As cláusulas *Else* e *ElseIf* são opcionais. Pode-se Ter tantas cláusulas *ElseIf* em um bloco *If* quantas desejar, mas nenhuma pode aparecer depois de uma cláusula *Else*. Instruções *If* em bloco podem ser embutidas, isto é, contidas uma dentro da outra.

Executando um *If* em bloco (segunda sintaxe), condição é testada. Se condição for *True*, as instruções seguintes a *Then* são executadas. Se condição for False, cada condição *ElseIf* (se houver) será por sua vez avaliada. Quando uma condição *True* for encontrada, as instruções imediatamente subseqüentes à *Then* associada são executadas. Se nenhuma das condições *ElseIf* for *True* (ou se não houver cláusulas *ElseIf*) as instruções subseqüentes a *Else* são executadas. Depois da execução das instruções subsequentes a *Then* ou *Else*, a execução continua com a instrução subsequente a *End If*.

Select Case

Executa um dos diversos grupos de instruções, dependendo do valor de uma expressão.

Sintaxe

```
Select Case ExpressãodeTeste
[Case ListadeExpressões-n
    [instruções-n]] ...
[Case Else
    [instruçõeselse]]
End Select
```

A sintaxe da instrução Select Case possui as partes a seguir:

Parte	Descrição
ExpressãodeTeste	Obrigatória. Qualquer expressão numérica ou expressão de seqüência.
ListadeExpressões-n	Obrigatória se aparecer uma Case. Lista delimitada de uma ou mais das seguintes formas: expressão, expressão To expressão, Is operadordeComparação expressão. A palavra-chave To especifica um intervalo de valores. Se você utilizar essa palavra-chave, o valor menor deverá aparecer antes de To. Utilize a palavra-chave Is com operadores de comparação (exceto Is e Like) para especificar um intervalo de valores. Se não for fornecida, a palavra-chave Is será inserida automaticamente.
Instruções-n	Opcional. Uma ou mais instruções são executadas se expressãodeTeste coincidir com qualquer parte de listadeExpressões-n.
InstruçõesElse	Opcional. Uma ou mais instruções são executadas se expressãodeTeste não coincidir com qualquer das cláusulas Case.

Comentários

Se *expressãodeTeste* coincidir com qualquer expressão Case *listadeExpressões*, as instruções subseqüentes àquela cláusula *Case* serão executadas até a próxima cláusula *Case* ou, para a última cláusula, até *End Select*. Então o controle passa para a instrução seguinte a *End Select*. Se *expressãodeTeste* coincidir com uma expressão *listadeExpressões* em mais de uma cláusula *Case*, somente as instruções subseqüentes à primeira coincidência serão executadas.

A cláusula *Case Else* é utilizada para indicar a *instruçõesElse* a ser executada se não for encontrada coincidência entre *expressãodeTeste* e uma *listadeExpressões* em qualquer das outras seleções de *Case*. Embora não seja necessário, é uma boa idéia Ter uma instrução *CaseElse* no seu bloco *Select Case* para manipular valores não previstos de *expressãodeTeste*. Se nenhuma *Case listadeExpressões* coincidir com *expressãodeTeste* e não houver instrução *Case Else*, a execução continua na instrução subseqüente a *End Select*.

Pode-se utilizar múltiplas expressões ou intervalos em cada cláusula *Case*. Por exemplo, a linha a seguir é válida.

```
Case 1 to 4, 7 to 9, 11, 13, Is > MaxNumber
```

Observação: O operador de comparação *Is* não é igual à palavra-chave *Is* utilizada na instrução *Select Case*.

Pode-se também especificar intervalos e várias expressões para seqüências de caracteres. No exemplo a seguir, *Case* coincide com seqüências que são exatamente iguais a tudo, seqüências que ficam entre nozes e sopa em ordem alfabética e o valor atual de *TesteItem*:

```
Case “tudo”, “nozes” to “sopa”, TestItem
```

Instruções *Select Case* podem ser embutidas. Cada instrução *Select Case* embutida deve Ter uma declaração *End Select* correspondente.

Utilize a instrução *Select Case* como uma alternativa a *ElseIf* nas instruções *If...Then...Else* ao comparar uma expressão a diversos valores diferentes. Embora as instruções *If...Then...Else* possam avaliar uma expressão diferente em cada instrução *ElseIf*, a instrução *Select Case* avalia somente uma expressão de cada vez, na parte superior da estrutura de controle.

No exemplo apresentado a seguir, a instrução *Select Case* avalia o argumento desempenho, que é passado para o procedimento. Observe que cada instrução *Case* pode conter mais de um valor, um intervalo de valores ou uma combinação de valores e operadores de comparação. A instrução *Case Else* opcional será executada caso a instrução *Select Case* não combine com um valor em umas das instruções *Case*.

```
Function Gratificação (desempenho, salário)
    Select Case Desempenho
        Case 1
            Gratificação = salário * 0.1
        Case 2, 3
            Gratificação = salário * 0,09
        Case 4 to 6
            Gratificação = salário * 0,07
        Case Is > 8
            Gratificação = 100
        Case Else
            Gratificação = 0
    End Select
End Function
```

Do ... Loop

Repete um bloco de instruções enquanto uma condição é *True* ou até que ela se torne *True*.

Sintaxe

```
Do [{While | Until } condição ]
Instruções
Exit Do
Instruções
Loop
```

Ou pode-se utilizar esta sintaxe:

```
Do
[instruções]
[Exit Do]
[Instruções]
Loop [{ While | Until } condição }
```

A sintaxe da instrução *Do Loop* possui as partes a seguir:

Parte	Descrição
Condição	Opcional. Expressão numérica ou expressão de seqüência que seja True ou False. Se condição

	for Null, condição é tratada como False.
Instruções	Uma ou mais instruções que são repetidas enquanto, ou até que, condição seja True.

Comentários: Qualquer número de instruções *Exit Do* pode ser colocado em qualquer lugar em *Do ... Loop* como meio alternativo para sair de um *Do ... Loop*. *Exit Do* é freqüentemente usado depois que alguma condição é avaliada, por exemplo, *If...Then*, caso em que a instrução *Exit Do* transfere o controle para a instrução imediatamente seguinte a *Loop*.

Quando utilizada com instruções *Do...Loop* embutidas, *Exit Do* transfere o comando para o loop que está embutido em um nível acima do loop em que ocorre *Exit Do*.

Este exemplo mostra como as instruções *Do...Loop* podem ser utilizadas. A instrução *Do...Loop* interna faz o loop 10 vezes, define o valor do sinalizador como *False* e sai prematuramente utilizando a instrução *Exit Do*. O loop externo sai imediatamente após a verificação do valor do sinalizador.

```
Dim Controle, Contador
Controle = True                    ‘ Inicializa variáveis
Contador = 0
Do                        ‘ Loop externo
    Do While contador < 20              ‘ Incrementa o contador
        Contador = Contador + 1    ‘ Se a condição for True
        If Contador = 10 Then      ‘ Define o valor do sinalizador
como False.
            Controle = False ‘ Sai do loop interno
            Exit Do
        End If
    Loop
Loop Until Controle = False             ‘ Sai do loop externo
imediatamente
```

For ... To ... Next

Pode-se utilizar instruções *For...Next* para repetir um bloco de instruções um determinado número de vezes. Os loops *For* utilizam uma variável de contador cujo valor é aumentado ou diminuído a cada repetição do loop.

O procedimento apresentado a seguir faz com que o computador emita 50 avisos sonoros. A instrução For especifica a variável de contador X e seus valores inicial e final. A instrução *Next* incrementa a variável de contador em uma unidade.

```
Sub Aviso Sonoro ()
    For X = 1 to 50
        Beep
    Next X
End Sub
```

A palavra-chave *Step* permite aumentar ou diminuir a variável de contador pelo valore especificado. No exemplo apresentado a seguir, a variável de contador J é incrementada por 2 todas as vezes que um loop se repete. Quando termina o loop, Total é a soma de 2, 4, 6, 8 e 10.

```
Sub TotaldeDois ()
    Total = 0
    For J = 2 to 10 step 2
        Total = Total + J
    Next J


    MsgBox "O total é " & Total
End Sub
```

Para diminuir a variável de contador, utilize um valor *Step* negativo e um valor final que seja inferior ao valore inicial. No exemplo apresentado a seguir, a variável de contador *meuNum* é reduzida em duas unidades a cada vez que o loop se repete. Quando o loop terminar, total será a soma de 16, 14, 12, 10, 8, 6, 4 e 2.

```
Sub NovoTotal ()
    Total = 0
For meuNum = 16 to 2 step –2
        Total = Total + meuNum
    Next meuNum
    MsgBox "O total é  " & total
End Sub
```

Observação: não é preciso incluir o nome da variável de contador depois da instrução Next. Nos exemplos apresentados acima o nome da variável foi incluído para deixá-los mais legíveis.

Pode-se sair de uma instrução *For...Next* antes que o comando atinja seu valor final utilizando a instrução *Exit For*. Por exemplo, quando ocorrer um erro, utilize a instrução *Exit For* no bloco de instruções *True* de uma instrução *If...Then...Else* ou uma instrução *Select case* que verifica especificamente o erro. Caso não ocorra o erro, então a instrução *If...Then...Else* será *False* e o loop prosseguirá normalmente.

For ... Each ... Next

Repete um grupo de instruções para cada elemento de uma matriz ou coleção

Sintaxe

```
For each elemento in Grupo
[Instruções]
[Exit For]
[Instruções]
Next [elemento]
```

A sintaxe da instrução *For...Each...Next* possui as partes a seguir:

Parte	Descrição
Elemento	Obrigatória. Variável utilizada para iterar através dos elementos da coleção ou matriz. Para as coleções, elemento pode ser somente uma variável Variant, uma variável de objeto genérica ou qualquer variável de objeto específica. Para as matrizes, elemento somente pode ser uma variável Variant.

Grupo	Obrigatória. Nome de uma coleção ou matriz de objetos (exceto matriz de tipos definidos pelo usuário)
Instruções	Opcional. Uma ou mais instruções que são executadas em cada item de um grupo.

Comentários

O bloco *For...Each* é inserido se houver pelo menos um elemento em grupo. Uma vez que o loop tenha sido inserido, todas as instruções do loop são executadas para o primeiro elemento do grupo. Se houver mais elementos em grupo, as instruções do loop continuam a ser executadas para cada elemento. Quando não houver mais elementos em grupo, o loop sai e execução continua com a instrução seguinte à instrução *Next*.

Qualquer número de instruções *Exit For* pode ser colocado em qualquer lugar do loop como um meio alternativo para sair. Muitas vezes, *Exit For* é utilizada depois de se avaliar alguma condição, por exemplo, *If... Then*, e transfere o controle para a instrução imediatamente seguinte a *Next*.

Pode-se embutir loops *For...Each....Next* colocando um dentro do outro. Entretanto, cada elemento do loop deve ser exclusivo.

Observação: se você emitir elementos em uma instrução *Next* a execução continua como se o elemento estivesse incluído. Se uma instrução *Next* for encontrada antes de sua instrução *For* correspondente, ocorrerá um erro.

Você não pode utilizar a instrução *For...Each...Next* com uma matriz de tipos definidos pelo usuário porque um *Variant* não pode conter um tipo de definido pelo usuário.

As instruções *For...Each...Next* repetem um bloco de instruções em cada objeto de uma coleção ou em cada elemento de uma matriz. O Visual Basic define automaticamente uma variável todas as vezes que o loop for executado. Por exemplo, o procedimento apresentado a seguir fecha todos os formulários, exceto o formulário que contém o procedimento que está sendo executado.

```
Sub FecharFormularios()
    For Each frm in Applications.Forms
        If frm.Caption <> Screen.ActiveForm.Caption Then frm.Close
    Next
End Sub
```

O código apresentado a seguir executa um loop através de todos os elementos de uma matriz e define o valor de cada um deles como o valor da variável de índice I.

```
Dim TestarMatriz(10) As Integer, I as Variant
For Each I in TestarMatriz
    TestarMatriz(I) = I
Next I
```

Executando um loop em um intervalo de células

Utilize um loop *For Each Next* para efetuar um loop nas células de um intervalo. O procedimento apresentado a seguir executa um loop no intervalo A1:D10 da Plan1 e define qualquer número cujo valor absoluto seja inferior a 0,01 como 0 (zero).

```
Sub ArredondarParaZero ()
    For Each meuObjeto in minhaColeção
        If Abs(meuObjeto.Value) < 0.01 Then meuObjeto.Value = 0
    Next
End Sub
```

Saindo de um loop *For...Each...Next* antes que ele tenha terminado

Pode-se sair de um loop For...Each...Next utilizando a instrução Exit For. Por exemplo, quando ocorre um erro, utilize a instrução Exit For no bloco de instruções True de uma instrução If...Then...Else ou uma instrução Select Case que verifica de forma específica o erro. Caso não ocorra o erro, a instrução If...Then...Else será False e o erro prosseguirá normalmente.

O exemplo apresentado a seguir testa a primeira célula no intervalo A1:B5 que não contenha um número. Se existir uma célula desse tipo, será exibida uma mensagem e Exit For sairá do loop.

```
Sub TestarNúmero()
    For Each meuObjeto in MinhaColeção
        If IsNumeric (meuObjeto.Value) = False then
            MsgBox "Objeto contém um valor não-numérico"
            Exit For
        End If
    Next
End Sub
```

With...End With

Executa uma série de instruções em um único objeto ou um tipo definido pelo usuário.

Sintaxe

```
With Objeto
[Instruções]
End With
```

A instrução With possui as partes a seguir:

Parte	Descrição
Objeto	Obrigatória. Nome de um objeto ou de um tipo definido pelo usuário
Instruções	Opcional. Uma ou mais instruções a serem executadas com Objeto

Comentários

A instrução *With* permite executar uma série de instruções com um objeto específico sem requalificar o nome do objeto. Por exemplo, para alterar diversas propriedades diferentes de um único objeto, coloque as instruções de atribuição de propriedades dentro da estrutura de controle *With*, referindo-se ao objeto uma só vez em lugar de fazê-lo em todas as atribuições de propriedade. O exemplo a seguir ilustra a utilização da instrução *With* para atribuir valores a diversas propriedades do mesmo objeto.

```
With MyLabel
    .Height = 2000
    .With = 2000
    .Caption = “Este é MyLabel”
End With
```

Observação: uma vez que um bloco *With* seja inserido, objeto não pode ser modificado. Como resultado, você não pode utilizar uma só instrução *With* para afetar objetos diferentes.

Pode-se embutir instruções *With* colocando blocos *With* um dentro do outro. Entretanto, em razão de os membros dos blocos externos de *With* ficarem mascarados dentro dos blocos internos, você deve proporcionar uma referência a objeto totalmente qualificada em um bloco *With* interno para qualquer membro de um objeto existente em um bloco externo.

Importante: Não salte para dentro e para fora de blocos *With*. Se as instrução em um bloco *With* forem executadas mas a instrução *With* ou *End With* não for, você poderá provocar erros ou comportamento imprevisível.

Este exemplo utiliza a instrução *With* para executar uma série de instruções em um único objeto. O objeto *meuObjeto* e as suas propriedades são nomes genéricos utilizados somente com finalidades ilustrativas.

```
With meuObjeto
    .Height = 100
    .Caption = “Alô mundo !!”
    With .Font
        .Color = Red
        .Bold = True
    End With
End With
```

Função MsgBox

Exibe uma mensagem em uma caixa de diálogo, aguarda que o usuário clique em um botão e retorna um integer que indica qual botão o usuário clicou.

Sintaxe

```
MsgBox (prompt[, buttons] [, title] [, helpfile, context])
```

A sintaxe da função MsgBox possui os argumentos nomeados a seguir:

Parte	Descrição
Prompt	Obrigatório. Expressão de sequência de caracteres exibida como mensagem na caixa de diálogo. O comprimento máximo é de aproximadamente 1024 caracteres, dependendo da largura dos caracteres utilizados. Se prompt consistir em mais de uma linha, você poderá separar as linhas utilizando um caracter de retorno de carro (Chr(13)), um caractere de alimentação de linha (Chr(10)) ou uma combinação de caracteres de retorno de carro e alimentação de linha (Chr(13)&Chr(10)) entre cada linha.
Buttons	Opcional. Expressão numérica que é a soma de valores que especifica o número e o tipo de botões a exibir, o estilo de ícone a utilizar, a identidade do botão padrão e a modalidade da caixa de mensagem. Se omitido, o valor padrão é 0.

Title	Opcional. Expressão de seqüência de caracteres exibida na barra de título da caixa de diálogo. Se você omitir title, o nome do aplicativo será inserido na barra de título.
Helpfile	Opcional. Expressão de seqüência de caracteres que identifica o arquivo de Ajuda a ser utilizado para fornecer a ajuda sensível ao contexto relativa a caixa de diálogo. Se helpfile for fornecido, context também deverá ser fornecido.
Context	Opcional. Expressão numérica que é o número de contexto da Ajuda atribuído ao tópico de Ajuda apropriado pelo autor da Ajuda.

Definições:

As definições do argumento *button* são:

Constante	Valor	Descrição
VbOkOnly	0	Exibe somente o botão Ok.
VbOkCancel	1	Exibe os botões Ok e Cancelar.
VbAbortRetryIgnore	2	Exibe os botões Abortar, Repetir e Ignorar
VbYesNoCancel	3	Exibe os botões Sim, Não e Cancelar
VbYesNo	4	Exibe os botões Sim e Não
VbRetryCancel	5	Exibe os botões Repetir e Cancelar
VbCritical	16	Exibe o ícone Mensagem Crítica
VbQuestion	32	Exibe o ícone Consulta de Aviso
VbExclamation	48	Exibe o ícone Mensagem de Aviso
VbInformation	64	Exibe o ícone Mensagem de Informação
VbDefaultButton1	0	O primeiro botão é o padrão.
VbDefaultButton2	256	O segundo botão é o padrão.
VbDefaultButton3	512	O terceiro botão é o padrão
VbDefaultButton4	768	O quarto botão é o padrão.
VbApplicationModal	0	Janela restrita do aplicativo; o usuário deve responder a caixa de mensagem antes de continuar o trabalho no aplicativo atual.
VbSystemModal	4096	Janela restrita do sistema; todos os aplicativos são suspensos até que o usuário responda a caixa de mensagem.

O primeiro grupo de valores (0-5) descreve o número e o tipo de botões exibidos na caixa de diálogo; o segundo grupo (16, 32, 48, 64) descreve o estilo do ícone; o terceiro grupo (0, 256, 512) determina qual botão é o padrão e o quarto grupo (0, 4096) determina a modalidade da caixa de mensagem. Quando estiver somando números para criar um valor final para o argumento *buttons*, utilize somente um número de cada grupo.

Observação: Estas constantes são especificadas pelo Visual Basic para Aplicativos. Como resultado, os nomes podem ser utilizados em qualquer lugar do seu código em vez dos valores reais.

Valores de retorno

Constante	Valor	Descrição
VbOk	1	Ok
VbCancel	2	Cancelar
VbAbort	3	Abortar
VbRetry	4	Repetir
VbIgnore	5	Ignorar
VbYes	6	Sim
VbNo	7	Não

Comentários: Quando *helpfile* e *context* são fornecidos, o usuário pode pressionar F1 para exibir o tópico da Ajuda correspondente ao *context*. Alguns aplicativos host, por exemplo, o Microsoft Excel, também adicionam automaticamente um botão Ajuda à caixa de diálogo.

Se a caixa de diálogo exibir um botão *Cancelar*, pressionar a tecla ESC terá o mesmo efeito que clicar em *Cancelar*. Se a caixa de diálogo contiver um botão Ajuda, será fornecida a ajuda sensível ao contexto relativa a caixa de diálogo. Entretanto nenhum valor será retornado até que um dos outros botões seja clicado.

Observação: Para especificar mais do que o primeiro argumento nomeado, pode-se utilizar *MsgBox* em uma expressão. Para omitir algum argumento posicional, você deve incluir o delimitador de vírgula correspondente.

Este exemplo utiliza a função *MsgBox* para exibir uma mensagem de erro crítico em uma caixa de diálogo com os botões Sim e Não. O botão Não é especificado como a resposta padrão. O valor retornado pela função *MsgBox* depende do botão escolhido pelo usuário. Este exemplo supõe que DEMO.HLP é um arquivo de ajuda que contém um tópico com um número de contexto da Ajuda igual a 1000.

```
Dim Msg, Estilo, Titulo, Ajuda, Ctxt, Resposta, MinhaSequencia
Msg = “Deseja continuar ?”
Estilo = vbYesNo + VbCritical + VbDefaultButton2
Titulo = “Demonstração do MsgBox”
Ajuda = “Demo.Hlp”
Ctxt = 1000

Resposta = MsgBox, (Msg, Estilo, Titulo, Ajuda, Ctxt)
If Resposta = VbYes then
    MinhaSequencia = “Sim”
Else
    MinhaSequencia = “Não”
End If
Função InputBox
```

Função InputBox

Exibe um aviso em uma caixa de diálogo, aguarda até que o usuário insira texto ou clique em um botão e retorna um *String* com o conteúdo da caixa de texto.

Sintaxe

```
InputBox (prompt[, title] [, default] [, xpos] [, ypos] [, helpfile,
context])
```

A sintaxe da função *InputBox* possui os argumentos nomeados a seguir:

Parte	Descrição
Prompt	Obrigatório. Expressão de sequência de caracteres exibida como mensagem na caixa de diálogo. O comprimento máximo de prompt é de aproximadamente 1024 caracteres, dependendo da largura dos caracteres utilizados. Se prompt consistir em mais de uma linha, você poderá separar as linhas utilizando um caracter de retorno de carro (Chr(13)), um caractere de alimentação de linha (Chr(10)) ou uma combinação de caracteres de retorno de carro e alimentação de linha (Chr(13)&Chr(10)) entre cada linha.
Title	Opcional. Expressão de seqüência de caracteres exibida na barra de título da caixa de diálogo. Se você omitir title, o nome do aplicativo será inserido na barra de título
Default	Opcional. Expressão de seqüência de caracteres exibida na caixa de texto como resposta padrão se nenhuma entrada for fornecida. Se você omitir default, a caixa de texto será exibida vazia.
Xpos	Opcional. Expressão númerica que especifica, em twips, a distância horizontal da extremidade esquerda da caixa de diálogo em relação a extremidade esquerda da tela. Se xpos for omitido, a caixa de diálogo será centralizada horizontalmente.
Ypos	Opcional. Expressão numérica que especifica, em twips, a distância vertical da extremidade superior da caixa de diálogo em relação ao topo da tela. Se ypos for omitido, a caixa de diálogo será posicionada na vertical a aproximadamente um terço da extremidade inferior da tela.
Helpfile	Opcional. Expressão de seqüência de caracteres que identifica o arquivo de Ajuda a ser utilizado para fornecer a ajuda sensível ao contexto relativa a caixa de diálogo. Se helpfile for fornecido, context também deverá ser fornecido.
Context	Opcional. Expressão númérica que é o número de contexto da Ajuda atribuído ao tópico de Ajuda apropriado pelo autor da Ajuda.

Comentários: Quando *helpfile* e *context* são fornecidos, o usuário pode pressionar F1 para exibir o tópico de ajuda que corresponde ao *context*. Alguns aplicativos host, por exemplo, o Microsoft Excel, também adicionam automaticamente um botão Ajuda à caixa de diálogo. Se o usuário clicar em *Ok* ou pressionar *Enter*, a função *InputBox* retornará o que houver na caixa de texto. Se o usuário clicar em *Cancelar*, a função retornará uma seqüência de caracteres de comprimento zero (“”)

Observação: Para especificar mais que o primeiro argumento nomeado, você deve utilizar *InputBox* em uma expressão. Para omitir alguns argumentos posicionais, você deve incluir o delimitador de vírgula correspondente.

Este exemplo mostra várias maneiras de utilizar a função *InputBox* para solicitar ao usuário que insira um valor. Se as posições x e y forem omitidas, a caixa de diálogo será automaticamente centralizada em relação aos respectivos eixos. A variável *meuValor* contém o valor inserido pelo usuário se o usuário clicar em *Ok* ou pressionar a tecla *Enter*. Se o usuário clicar em *Cancelar*, será retornada uma seqüência de comprimento zero.

```
Dim Mensagem, Titulo, Padrão, MeuValor
Mensagem = “Insira um valor entre 1 e 3”
Título = “Demonstração da caixa de entrada”
Padrão = “1”
‘ Exibe a mensagem, o título e o valor padrão
MeuValor = InputBox (Mensagem, Título, Padrão)
‘Utiliza o arquivo de Ajuda e o contexto. O botão Ajuda é adicionado
automaticamente
MeuValor = InputBox (Mensagem, Título, , , , “DEMO.HLP”, 10)
‘ Exibe a caixa de diálogo na posição 100, 100
MeuValor = InputBox (Mensagem, Titulo, Padrão, 100, 100)
```

Regras de nomenclatura de objetos

Utilize as seguintes regras quando nomear procedimentos, constantes, variáveis e argumentos em um módulo do Visual Basic:

Você deve utilizar uma letra como primeiro caractere.

Você não pode utilizar um espaço, ponto, ponto de exclamação ou os caracteres @, &, $, # no nome.

O nome pode ter no máximo 255 caracteres de comprimento

Geralmente você não deve utilizar nomes iguais às funções, instruções e métodos do Visual Basic. Pode-se sombrear as mesmas palavras chave na linguagem. Para utilizar uma função, instrução ou método intrínseco que entre em conflito com um nome atribuído, você deve identifica-lo explicitamente. Proceda o nome da função, instrução ou método intrínseco com o nome da biblioteca de tipos a que está associado. Por exemplo, se você tiver uma variável chamada *Left*, só poderá invocar a função *Left* utilizando *VBA.Left.*

Você não pode repetir nomes dentro do mesmo nível de escopo. Por exemplo, você não pode declarar duas variáveis com o nome idade dentro do mesmo procedimento. No entanto, é possível declarar uma variável privada com o nome idade e uma variável em nível de procedimento com o nome idade dentro do mesmo módulo.

Observação: O Visual Basic não faz distinção entre maiúsculas e minúsculas, mas preserva a capitalização na instrução em que o nome for declarado.

Resumo dos operadores

Operadores	Descrição
Operadores aritméticos	Operadores utilizados para efetuar cálculos matemáticos: ^, *, /, \, Mod, +, -
Operadores de comparação	Operadores utilizados para efetuar comparações >, <, <>, >=, <=, IS, LIKE
Operadores de concatenação	Operadores utilizados para combinar seqüências de caracteres. &, +
Operadores lógicos	And, Eqv, Imp, Not, Or, Xor

Resumo das palavras-chave de datas e horas

Ação	Palavras-chave

Obter a data ou a hora atual	Date, Now, Time
Executar cálculos com datas	DateAdd, DateDiff, DatePart
Retornar uma data	DateSerial, DateValue
Retornar uma hora	TimeSerial, TimeValue
Definir a data ou a hora	Date, Time
Medir o tempo de um processo	Timer

Resumo das palavras-chave de fluxo de controle

Ação	Palavras-chave
Ramificar	GoSub...Return, GoTo, On Error, On...GoSub, On...GoTo
Sair ou fazer pausa no programa	DoEvents, End, Exit, Stop
Executar loop	Do...Loop, For...Next, For Each... Next, While...Wend, With
Tomar decisões	Choose, If...Then...Else, Select Case, Switch
Usar procedimentos	Call, Function, Property Get, Property Let, Property Set, Sub

Resumo das palavras-chave de manipulação de seqüências de caracteres

Ação	Palavras-chave
Comparar duas seqüências de caracteres	StrComp
Converter seqüências de caracteres	StrConv
Converter em minúsculas ou maiúsculas	Format, LCase, UCase
Criar seqüência de caracter repetido	Space, String
Encontrar o tamanho de uma seqüência de caracteres	Len
Formatar uma seqüência de caracteres	Format
Justificar uma seqüência de caracteres	LSet, RSet
Manipular seqüências de caracteres	InStr, Left, LTrim, Mid, Right, RTrim, Trim
Definir regra de comparação de seqüências de caracteres	Option Compare
Trabalhar com valores ASCII e ANSI	Asc, Chr

Resumo das palavras-chave de tipo de dados

Ação	Palavras-chave
Converter entre tipos de dados	CBool, CByte, CCur, CDate, CDbl, CDec, CInt, CLng, CSng, CStr, CVar, CVErr, Fix, Int

Definir tipos de dados intrínsecos	Boolean, Byte, Currency, Date, Double, Integer, Long, Object, Single, String, Variant (default)
Verificar tipos de dados	IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject

Resumo das palavras-chave financeiras

Ação	Palavras-chave
Calcular depreciação	DDB, SLN, SYD
Calcular valor futuro	FV
Calcular taxa de juros	Rate
Calcular taxa de retorno interna	IRR, MIRR
Calcular número de períodos	Nper
Calcular pagamentos	IPmt, Pmt, PPmt
Calcular valor presente	NPV, PV

Resumo das palavras chave de variáveis e constantes

Ação	Palavras-chave
Atribuir valor	Let
Declarar variáveis ou constantes	Const, Dim, Private, Public, New, Static
Declarar módulo como privado	Option Private Module
Obter informações sobre uma variante	IsArray, IsDate, IsEmpty, IsError, IsMissing, IsNull, IsNumeric, IsObject, TypeName, VarType
Referir-se ao objeto atual	Me
Requerer declarações de variável explicita	Option Explicit
Definir tipo de dados padrão	Deftipo

Trabalhando com dados da planilha

Uma tarefa comum ao usar o Visual Basic é especificar uma célula ou intervalo de células e, em seguida, fazer algo com elas, como inserir uma fórmula ou alterar o formato. Geralmente, pode-se fazer isso em uma instrução que identifique o intervalo e também altere uma propriedade ou aplique um método.

Um objeto *Range* no Visual Basic pode ser uma única célula ou um intervalo de células. Os tópicos seguintes mostram as maneiras mais comuns de identificar e trabalhar com objetos *Range*.

Referindo-se a célula usando notação A1

Pode-se referir-se a uma célula ou intervalo de células no estilo de referência A1 usando o método Range. O procedimento Sub seguinte altera o formato das células A1:D5 para negrito:

```
Sub FormatarIntervalo()
    Workbooks("Pasta1").Sheets("Plan1").Range("A1:D5") _
        .Font.Bold = True
End Sub
```

A tabela seguinte ilustra algumas referências em estilo A1 usando o método *Range*:

Referência	Significado
Range("A1")	Célula A1
Range("A1:B5")	Células de A1 até B5
Range("C5:D9,G9:H16")	Seleção de várias áreas
Range("A:A")	Coluna A
Range("1:1")	Linha 1
Range("A:C")	Colunas de A até C
Range("1:5")	Linhas de 1 até 5
Range("1:1,3:3,8:8)	Linhas 1, 3 e 8
Range("A:A,C:C,F:F")	Colunas A, C e F

Referindo-se a células usando números de índice

Pode-se usar a propriedade *Cells* para referir-se a uma única célula usando números de índices de linhas e coluna, Essa propriedade retorna um objeto *Range* representando uma única célula. No exemplo seguinte, *Cells(6,1)* retorna a célula A6 de Plan1. Em seguida a propriedade *Value* é definida como 10.

```
Sub InserirValor()
    Worksheets("Plan1").Cells(6,1).Value = 10
End Sub
```

A propriedade *Cells* funciona bem para loop através de um intervalo de células porque pode-se substituir os números de índice por variáveis, conforme mostrado no exemplo seguinte:

```
Sub Circular()
Dim Contador as Integer
For Contador = 1 to 20
    Worksheets("Plan1").Cells(Contador,3).Value = Contador
Next Contador
End Sub
```

Observação: Se você desejar alterar as propriedades ou aplicar um método a todo um intervalo de células de uma só vez, use a propriedade Range. Para obter maiores informações, consulte Referir-se a células usando notação A1.

Referindo-se a linhas e colunas

Use a propriedade *Rows* ou a propriedade *Columns* para trabalhar com linhas ou colunas inteiras. Essas propriedades retornam um objeto *Range* representado um intervalo de células. No exemplo seguinte, *Rows(1)* retorna a linha 1 em Plan1. Em seguida, a propriedade *Bold* do objeto *Font* do intervalo é definida como *True*.

```
Sub LinhaComNegrito()
    Worksheets("Plan1").Rows(1).Font.Bold = True
End Sub
```

A tabela seguinte ilustra algumas referências de linha e coluna usando as propriedades *Rows* e *Colums*.

Referência	Significado
Rows(1)	Linha 1
Rows	Todas as linhas da planilha
Columns(1)	Coluna 1
Columns("A")	Coluna 1
Columns	Todas as colunas da planilha

Para trabalhar com várias linhas ou colunas ao mesmo tempo, crie uma variável de objeto e use o método *Union*, combinando várias chamadas à propriedade *Rows* ou *Columns*. O exemplo seguinte altera para negrito o formato das linhas 1, 3 e 5 na planilha 1 da pasta de trabalho ativa.

```
Sub VariasLinhas()


Dim MinhaUniao as Range
    Worksheets("Plan1").Activate
    Set MinhaUniao = Union(Rows(1), Rows(3), Rows(5))
    MinhaUniao.Font.Bold = True
End Sub
```

Referindo-se a células usando notação de atalho

Pode-se usar o estilo de referência A1 ou um intervalo nomeado entre colchetes com um atalho para a propriedade *Range*. Você não precisa digitar a palavra "*Range*" nem usar aspas, conforme mostrado nos exemplos seguintes.

```
Sub LimparIntervalo()
    Worksheets("Plan1").[A1:B5].ClearContents
End Sub
Sub DefinirValor()
    [MeuIntervalo].Value = 30
End Sub
```

Referindo-se a intervalos nomeados

Os intervalos são mais fáceis de identificar por nome do que por notação A1. Para nomear um intervalo selecionado, clique na caixa de nome na extremidade esquerda da barra de fórmulas, digite um nome e, em seguida, pressione ENTER.

O exemplo seguinte refere-se ao intervalo chamado "*MeuIntervalo*" na pasta de trabalho chamada "MinhaPasta.xls".

```
Sub FormatarIntervalo()
    Range("MinhaPasta.xls!MeuIntervalo").Font.Italic = True
End Sub
```

O exemplo seguinte refere-se ao intervalo específico de planilha chamado "Plan1!Vendas" na pasta de trabalho chamada "Relatório.xls".

```
Sub Formatar Vendas()
    Range("[Relatório.xls]Plan1!Vendas").BorderAround weight:=xlthin
End Sub
```

Para selecionar um intervalo nomeado, use o método *GoTo*, o qual ativa a pasta de trabalho e a planilha e, em seguida, seleciona o intervalo.

```
Sub LimparIntervalo()
    Application.GoTo Reference:=Range("MinhaPasta.xls!MeuIntervalo")
    Selection.ClearContents
End Sub
```

O exemplo seguinte mostra como o mesmo procedimento seria escrito para a pasta de trabalho ativa:

```
Sub LimparIntervalo()
    Application.GoTo Reference:=Range("MeuIntervalo")
    Selection.ClearContents
End Sub
```

Loop através de células em um intervalo nomeado

O exemplo seguinte faz um loop através de cada célula de um intervalo nomeado usando um loop *For Each...Next*. Se o valor de qualquer célula do intervalo exceder o valor de limite, a cor da célula será alterada para amarelo.

```
Sub AplicarCor()
    Const Limite as Integer = 25
    For Each C in Range ("Meu Intervalo")
        IF C.Value > Limite then
            C. Interior.ColorIndex = 27
        End If
    Next C
End Sub
```

Referindo-se a células em relação a outras células

Uma maneira comum de trabalhar com uma célula em relação a uma outra célula é usar a propriedade *Offset*. No exemplo seguinte, o conteúdo da célula que se encontra uma linha abaixo e a três coluna da célula ativa na planilha ativa é formatado com sublinhado duplo.

```
Sub Sublinhar()
    ActiveCell.Offset(1,3).Font.Underline = xlDouble
End Sub
```

Observação: Pode-se gravar macros que usem a propriedade *Offset* em vez de referências absolutas. No menu Ferramentas, aponte para Gravar macro e, em seguida, clique em Usar referências relativas.

Para fazer um loop através de um intervalo de células, use uma variável com a propriedade *Cells* em um loop. O exemplo seguinte preenche as 20 primeiras células da terceira coluna com valores entre 5 e 100, incrementados por 5. A variável contador é usada como índice de linha para a propriedade *Cells*.

```
Sub Circular()
    Dim Contador as Integer
    For Contador = 1 to 20
        Worksheets("Plan1").Cells(Contador, 3).Value = Contador*5
    Next Contador
End Sub
```

Referindo-se a células usando um objeto Range

Quando você define uma variável de objeto para um objeto Range, pode-se facilmente manipular o intervalo usando o nome da variável.

O procedimento seguinte cria a variável de objeto MeuIntervalo e, em seguida, atribui a variável ao intervalo A1:D5 de Plan1 na pasta de trabalho ativa. Instruções subsequentes modificam propriedades do intervalo substituindo o objeto de intervalo pelo nome da variável.

```
Sub Aleatorizar()
    Dim MeuIntervalo as Range
    Set MeuIntervalo = Worksheets("Plan1").Range("A1:D5")
    MeuIntervalo.Formula = "=RAND()"
    MeuIntervalo.Font.Bold = True
End Sub
```

Referindo-se a todas as células da planilha

Quando você aplica a propriedade *Cells* a uma planilha sem especificar um número de índice, o método retorna um objeto Range representando todas as células da planilha. O procedimento Sub seguinte limpa o conteúdo de todas as células de Plan1 na pasta de trabalho ativa.

```
Sub LimparPlanilha()
    Worksheets("Plan1").Cells.ClearContents.
End Sub
```

Loop através de um intervalo de células

Ao usar o Visual Basic, você freqüentemente precisa executar o mesmo bloco de instruções em cada célula de um intervalo de células. Para fazer isso, você combina uma instrução de loop com um ou mais métodos para identificar cada célula, uma de cada vez, e executa a operação.

Uma maneira de fazer loop através de um intervalo é usar o loop *For...Next* com a propriedade *Cells*. Usando a propriedade *Cells*, pode-se substituir o contador do loop (ou outras variáveis ou expressões) pelos números de índice das células. No exemplo seguinte, a variável Contador é substituída pelo índice de linha. O procedimento faz um loop através de um intervalo C1:C20, definindo como 0 (zero) qualquer número cujo valor absoluto seja menor que 0,01.

```
Sub ArredondarParaZero()
    For Contador = 1 to 20
        Set CelulaAtual = Worksheets("Plan1").Cells(contador,3)
        If Abs(CelulaAtual.Value) < 0,01 Then CelulaAtual.Value = 0
    Next Contador
End Sub
```

Uma outra maneira mais fácil de se fazer um loop através de um intervalo é usar um loop *For Each...Next* com a coleção de células retornada pelo método *Range*. O Visual Basic define automaticamente uma variável de objeto para a próxima célula cada vez que o loop é executado. O procedimento seguinte faz um loop através do intervalo A1:D10, definindo como 0 (zero) qualquer número cujo valor absoluto seja menor que 0,01.

```
Sub ArredondarParaZero()
    For Each C in Worksheets("Plan1").Range("A1:D10").Cells
        If Abs(C.Value) < 0,01 Then C.Value = 0
    Next
End Sub
```

Se você não souber os limites do intervalo pelo qual deseja fazer o loop pode-se usar a propriedade *CurrentRegion* para retornar o intervalo que envolve a célula ativa. Por exemplo, o procedimento seguinte, quando executado de uma planilha, faz um loop através do intervalo que envolve a célula ativa, definindo como 0 (zero) qualquer número cujo valor seja menor que 0,01.

```
Sub ArredondarParaZero()
    For Each C in ActiveCell.CurrentRegion.Cells
        If Abs(C.Value) < 0,01 Then C.Value = 0
    Next
End Sub
```

Selecionar e ativar células

Quando você trabalho com o Microsoft Excel, você geralmente seleciona uma célula ou células e, em seguida, efetua uma ação, como formatar as células ou inserir valores nelas. No Visual Basic, normalmente não é necessário selecionar células antes de modificá-las.

Por exemplo, se você desejar inserir uma fórmula na célula D6 usando o Visual Basic, você não terá que selecionar o intervalo D6. Você precisa apenas retornar o objeto *Range* e, em seguida, definir a propriedade *Formula* com a fórmula desejada, conforme mostrada no exemplo seguinte.

```
Sub InserirFormula()
    Worksheets("Plan1").Range("D6").Formula = "=SUM(D2:D5)"
End Sub
```

Para exemplos de uso de outros métodos para controlar células sem selecioná-las, consulte Como referir-se a células e intervalos.

Usar o método Select e a propriedade Selection

O método *Select* ativa planilhas e objetos em planilhas; a propriedade *Selection* retorna um objeto representando a seleção atual na planilha ativa da pasta de trabalho ativa. Antes de você poder usar com êxito a propriedade *Selection*, você precisa ativar uma pasta de trabalho, ativar ou

selecionar uma planilha e, em seguida, selecionar um intervalo (ou outro objeto) usando o método *Select*.

O gravador de macro costuma criar macros que usam o método *Select* e a propriedade *Selection*. O procedimento *Sub* seguinte foi criado pelo uso do gravador de macro e ilustra como *Select* e *Selection* funcionam juntas.

```
Sub Macro1()
    Sheets("Plan1").Select
    Range("A1").Select
    ActiveCell.FormulaR1C1 = "Nome"
    Range("B1").Select
    ActiveCell.FormulaR1C1 = "Endereço"
    Range("A1:B1").Select
    Selection.Font.Bold = True
End Sub
```

O exemplo seguinte realiza a mesma tarefa sem ativar nem selecionar a planilha ou as células.

```
Sub Legendas()
    With Worksheets("Plan1")
        .Range("A1") = "Nome"
        .Range("B1") = "Endereço"
        .Range("A1:B1").Font.Bold = True
    End With
End Sub
```

Selecionar células na planilha ativa

Se você usa o método *Select* para selecionar células, esteja ciente de que *Select* só funciona na planilha ativa. Se você executar o seu procedimento Sub a partir do módulo, o método *Select* falhará a menos que o seu procedimento ative a planilha antes de usar o método *Select* em um intervalo de células. Por exemplo, o procedimento seguinte copia uma linha de Plan1 para Plan2 na pasta de trabalho ativa.

```
Sub CopiarLinha()
    Worksheets("Plan1").Rows(1).Copy
    Worksheets("Plan2").Select
    Worksheets("Plan2").Rows(1).Select
    Worksheets("Plan2").Paste
End Sub
```

Ativar uma célula dentro de uma seleção

Pode-se usar o método Activate para ativar uma célula dentro de uma seleção. Só pode haver uma célula ativa, mesmo quando um intervalo de células é selecionado. O procedimento seguinte seleciona um intervalo e, em seguida, ativa uma célula dentro do intervalo sem alterar a seleção.

```
Sub TornarAtiva()
    Worksheets("Plan1").Activate
    Range("A1:D4").Select
    Range("B2").Activate
End Sub
```

Trabalhar com a célula ativa

A propriedade *ActiveCell* retorna um objeto *Range* representando a célula que está ativa. Pode-se aplicar qualquer das propriedades ou métodos de um objeto *Range* à célula ativa, como no exemplo seguinte.

```
Sub DefinirValor()
    Worksheets("Plan1").Activate
    ActiveCell.Value = 35
End Sub
```

Observação: Você só pode trabalhar com a célula ativa quando a planilha na qual ela se encontra é a planilha ativa.

Mover a célula ativa

Pode-se usar o método *Activate* para designar qual célula é a célula ativa. Por exemplo, o procedimento seguinte torna B5 a célula ativa e, em seguida, a formata com negrito.

```
Sub DefinirAtiva()
    Worksheets("Plan1").Activate
    Worksheets("Plan1").Range("B5").Activate
    ActiveCell.Font.Bold = True
End Sub
```

Observação: Para selecionar um intervalo de células use o método *Select*. Para tornar uma única célula a célula ativa, use o método *Activate*.

Pode-se usar a propriedade *Offset* para mover a célula ativa. O procedimento seguinte insere texto na célula ativa do intervalo selecionado e, em seguida, move a célula ativa uma célula para a direita sem alterar a seleção.

```
Sub MoverAtiva()
Worksheets("Plan1").Activate
Range("A1:D10").Select
ActiveCell.Value = "Totais Mensais"
ActiveCell.Offset(0,1).Activate
End Sub
```

Selecionar as células ao redor da célula ativa

A propriedade *CurrentRegion* retorna um intervalo de células delimitado por linhas e colunas em branco. No exemplo seguinte, a seleção é expandida para incluir as células adjacentes à célula ativa que contenham dados. Em seguida, esse intervalo é formatado com o formato *Moeda*.

```
Sub Regiao()
    Worksheets("Plan1").Activate
    ActiveCell.CurrentRegion.Select
    Selection.Style = "Currency"
End Sub
```

Criando uma interface com o usuário

Tal como pode-se adicionar controles ActiveX a caixas de diálogo personalizadas, pode-se adicionar controles diretamente a um documento quando desejar fornecer uma maneira sofisticada do usuário interagir diretamente com sua macro sem a distração das caixas de diálogo. Use o procedimento seguinte para adicionar controles ActiveX a seu documento. Para obter informações mais específicas sobre uso de controles ActiveX no Microsoft Excel, consulte Usar controles ActiveX em planilhas.

1. Adicionar controles ao documento

   Para adicionar controles a um documento, exiba a Caixa de Ferramentas de Controle, clique no controle que você deseja adicionar e, em seguida, clique no documento. Arraste uma alça de ajuste do controle até o contorno do controle Ter o tamanho e a forma desejados. Arrastar um controle (ou vários controles “agrupados”) do formulário de volta para a Caixa de Ferramentas de Controle cria um modelo desse controle, o qual pode ser reutilizado. Isso é um recurso útil para implementação de uma aparência e um modo de operação padronizados em seus aplicativos.

2. Definir propriedades de controle

   Pode-se definir algumas propriedades de controle quando da criação (antes de qualquer macro ser executada). Em modo de criação, clique com o botão direito do mouse em um controle e clique em Propriedades. Os nomes das propriedades são mostrados na coluna esquerda da janela e os valores das propriedades na coluna direita. Você define o valor de uma propriedade inserindo o novo valor à direita do nome da propriedade.

3. Inicializar os controles

   Pode-se inicializar controles em tempo de execução usando código do Visual Basic em uma macro. Por exemplo, pode-se preencher uma caixa de listagem, definir valores de texto ou definir botões de opção.

   O exemplo seguinte usa o método *AddItem* para adicionar dados a uma caixa de listagem. Em seguida, ele define o valor de uma caixa de texto e exibe o formulário.

```
Private Sub ObterNomeDoUsuario()
    With FormDoUsuario
         .lstRegioes.AddItem “Norte”
         .lstRegioes.AddItem “Sul”
         .lstRegioes.AddItem “Leste”
         .lstRegioes.AddItem “Oeste”
         .txtCódigodoVendedor.Text = “00000”
         .Show
    End With
End Sub
```

Você também pode usar código no evento *Initialize* de um formulário para definir valores iniciais para controles no formulário. Uma vantagem da definição de valores iniciais para controles no evento *Initialize* é que o código de inicialização permanece com o formulário. Pode-se copiar o formulário para um outro projeto com todas as inicializações.

```
Private Sub UserForm_Initialize)
    FormDoUsuario.lstNomes.AddItem “Teste Um”
    FormDoUsuario.lstNomes.AddItem “Teste Dois”
```

```
    FormDoUsuario.txtNomeDoUsuario.Text = “Nome Padrão”
End Sub
```

5. Escrever procedimentos de evento

   Os *UserForms* e os controles têm um conjunto predefinido de eventos. Por exemplo, um botão de comando tem um evento *Click* que ocorre quando o usuário clica no botão de comando, e os *UserForms* têm um evento *Initialize* que é executado quando o formulário é carregado. Pode-se escrever procedimentos de evento que sejam executados quando os eventos ocorrem.

   Após adicionar controles a sua caixa de diálogo ou documento, você adiciona procedimentos de evento para determinar como os controles responderão a ações do usuário.

   Para escrever um controle ou procedimento de evento de formulário, abra um módulo clicando duas vezes no formulário ou controle e selecione o evento a partir da caixa de listagem drop-down Procedimento.

   Os procedimentos de evento incluem o nome do controle. Por exemplo, o nome do procedimento de evento Click de um botão de comando chamado Comando1 é Comando1_Click.

   Se você adicionar código a um procedimento de evento e, em seguida, alterar o nome do controle, o seu código permanecerá nos procedimentos com o nome anterior.

   Por exemplo, suponha que você adicione código ao evento Click de Comando1 e, em seguida, renomeie o controle para Comando2. Quando você clicar duas vezes em Comando2, você não verá código algum no procedimento de evento Click. Você terá que mover o código de Comando1_Click para Comando2_Click.

   Para simplificar o desenvolvimento, é uma boa prática nomear corretamente seus controles antes de escrever código.

6. Usar valores de controle quando o código está sendo executado

   Algumas propriedades podem ser definidas em tempo de execução. O exemplo seguinte define a propriedade Text de uma caixa de texto como “Olá”

   ```
   CaixaDeTexto1.Text = “Olá”
   ```

   Os dados inseridos em um formulário por um usuário são perdidos quando o formulário é fechado. Quando você retorna os valores dos controles de um formulário após o formulário ter sido descarregado, você obtém os valores iniciais dos controles em vez dos valores inseridos pelo usuário.

   Se você deseja salvar os dados inseridos em um formulário, pode-se salvar as informações em variáveis a nível de módulo enquanto o formulário ainda está sendo executado. O exemplo seguinte exibe um formulário e salva os dados do formulário.

```
‘ Código no módulo para declarar variáveis públicas
Public strRegiao As String
Public intCodigoDoVendedor as Integer
Public blnCancelado as Boolean

‘ Código no formulário
```

```
Private Sub cmdCancelar_Click()
    Modulo1.blnCancelado = True
    Unload Me
End Sub

Private Sub cmdOk_Click()
    ‘ Salvar dados
    intCodigoDoVendedor = txtCodigoDoVendedor.Text
    strRegiao = lstRegioes.List (lstRegioes.ListIndex)
    Modulo1.blnCancelado = False
    Unload Me
End Sub

Private Sub UserForm_Initialize ()
    Modulo1.blnCancelado = True
End Sub

‘ Código em módulo para exibir formulário
Sub IniciarFormulárioVendedores()
    frmVendedores.Show
    if blnCancelado = True Then
        MsgBox “Operação Cancelada !”, vbExclamation
    Else
        MsgBox “O código do Vendedor é : “ & _
            IntCodigoDoVendedor & Chr$(13) & _
            “A região é : “ & strRegiao
    End If
End Sub
```

Erros e tratamentos de erros

Quando você está programando um aplicativo, você precisa levar em consideração o que acontece quando ocorre um erro. Um erro pode ocorrer no seu aplicativo por duas razões. Primeiro, quando alguma condição durante a execução do aplicativo faz com que o código, que na ausência dessa situação seria válido, apresenta uma falha. Por exemplo, se o seu código tentar abrir uma tabela que o usuário excluiu, ocorrerá um erro. Segundo, quando o código contém lógica incorreta que o impede de fazer o que você pretendia. Por exemplo, ocorre um erro se o seu código tentar dividir um valor por zero.

Se você não implementou nenhum tratamento de erros, o Visual Basic interromperá a execução e exibirá uma mensagem de erro, quando ocorrer um erro no seu código. O usuário do seu aplicativo provavelmente se sentirá confuso e frustado quando isso acontecer. Você poderá prevenir muitos problemas incluindo rotinas minuciosas de tratamento de erros no seu código, para manipular qualquer erro que possa ocorrer.

Ao adicionar tratamento de erros em um procedimento, você deverá levar em consideração como o procedimento irá direcionar a execução quando ocorrer um erro. O primeiro passo no direcionamento da execução para uma rotina de tratamento de erros é ativar a rotina por meio da inclusão de alguma forma da instrução *On Error* no procedimento. A instrução *On Error* direciona a execução na eventualidade de um erro. Se não houver uma instrução *On Error*, o Visual Basic simplesmente interromperá a execução e exibirá uma mensagem de erro.

Quando ocorre um erro em um procedimento que tem uma rotina de tratamento de erros ativada, o Visual Basic não exibe a mensagem normal de erro. Em vez disso, ele direciona a execução para uma rotina de tratamento de erros, caso exista. Quando a execução é passada para uma rotina de tratamento de erros habilitada, esta rotina torna-se ativa. Dentro da rotina de tratamento de erros, pode-se verificar o tipo de erro ocorrido e tratá-lo da maneira que escolher. O Microsoft Excel oferece dois objetos que contêm informações sobre erros ocorridos: o objeto *Err* do Visual Basic e o objeto *Err* do DAO.

Direcionando a execução quando ocorre um erro

Uma rotina de tratamento de erros especifica o que acontece dentro de um procedimento quando ocorre um erro. Por exemplo, pode-se desejar que o procedimento seja finalizado se ocorrer um certo erro, ou que seja corrigida a condição que causou o erro e a execução continue. As instruções *On Error* e *Resume* determinam como prosseguirá a execução na eventualidade de um erro.

A instrução On Error

Esta instrução ativa ou desativa uma rotina de tratamento de erros. Se uma rotina de tratamento de erros estiver ativada, a execução passará para esta rotina quando ocorrer um erro.

Existem três formas da instrução *On Error*: *On Error GoTo rótulo*, *On error GoTo 0* e *On Error Resume Next.* A instrução *OnError GoTo rótulo* ativa uma rotina de tratamento de erros a começar da linha na qual se encontra a instrução. Você deve ativar a rotina de tratamento de erros antes da primeira linha em que um erro possa ocorrer. Quando o rotina de tratamento de erros está ativa e ocorre um erro, a execução passa para a linha especificada pelo argumento rótulo.

A linha especificada pelo argumento rótulo deve ser o início da rotina de tratamento de erros. Por exemplo. O procedimento a seguir especifica que, se ocorrer um erro, a execução passará a linha intitulada Error_PodeCausarUmErro.

```
Function PodeCausarUmErro()
    ‘ Ativa rotina de tratamento de erros
    On Error GoTo Error_PodeCausarUmErro

    ‘ Inclua aqui código que possa gerar um erro
    ...
Erro_PodeCausarUmErro:
    ‘ Inclua aqui código para tratamento de erros.
    ...
End Function
```

A instrução *On Error GoTo 0* desativa o tratamento de erros dentro de um procedimento. Ela não especifica a linha 0 como o início do código de tratamento de erros, mesmo que o procedimento contenha uma linha numerada como 0. Se não houver nenhuma instrução *On Error GoTo 0* no seu código, a rotina de tratamento de erros é desativada automaticamente quando o procedimento for executado até o fim. A instrução *On Error GoTo 0* redefine as propriedades do objeto *Err*, tendo o mesmo efeito do método *Clear* o objeto *Err*.

A instrução *On Error Resume Next* ignora a linha que causa um erro e direciona a execução para a linha seguinte àquela que causou o erro. A execução não é interrompida. Pode-se utilizar a instrução *On Error Resume Next* se quiser verificar as propriedades do objeto *Err* imediatamente após uma linha na qual você prevê que ocorrerá um erro, e tratar o erro dentro do procedimento, ao invés de fazê-lo dentro de uma rotina de tratamento de erros.

A instrução Resume

A instrução *Resume* devolve a execução ao corpo do procedimento, a partir de uma rotina de tratamento de erros. Pode-se incluir uma instrução *Resume* dentro de uma rotina de tratamento de erros se desejar que a execução continue em um determinado ponto de um procedimento. Entretanto, a instrução *Resume* não é necessária; pode-se também finalizar o procedimento depois da rotina de tratamento de erros.

Existem três formas da instrução *Resume*. A instrução *Resume* ou *Resume 0* retorna a execução para a linha na qual ocorreu o erro. A instrução *Resume Next* retorna a execução para a linha imediatamente seguinte aquela na qual ocorreu o erro. A instrução *Resume rótulo* retorna a execução para a linha especificada pelo argumento rótulo. O argumento *rótulo* deve indicar um rótulo ou número de linha.

Em geral, você utilizar a instrução *Resume* ou *Resume 0* quando o usuário precisa fazer uma correção. Por exemplo, se você solicita ao usuário o nome de uma tabela a ser aberta e o usuário insere o nome de uma tabela que não existe, pode-se fazer nova solicitação e continuar a execução na instrução que causou o erro.

Você utiliza a instrução *Resume Next* quando o seu código corrige o erro dentro de uma rotina de tratamento de erros e você deseja continuar a execução sem executar novamente a linha que causou o erro. Você utiliza a instrução *Resume rótulo* quando deseja continuar a execução em outro ponto do procedimento, especificado pelo argumento *rótulo*. Por exemplo, pode ser conveniente continuar a execução em uma rotina de saída, conforme descrito na próxima seção.

Saindo de um procedimento

Quando você inclui uma rotina de tratamento de erros em um procedimento, você deve também incluir uma rotina de saída, para que a rotina de tratamento de erros só seja executada se ocorrer um erro. Pode-se especificar uma rotina de saída com um rótulo de linha, da mesma maneira que especifica uma rotina de tratamento de erros.

Por exemplo, pode-se adicionar uma rotina de saída ao exemplo da seção anterior. Se não ocorrer um erro, a rotina de saída será executada depois do corpo do procedimento. Se ocorrer um erro, a execução passará para a rotina de saída após execução da rotina de tratamento de erros. A rotina de saída contém uma instrução *Exit*.

```
Function PodeCausarUmErro()
    ‘ Ativa rotina de tratamento de erros
    On Error GoTo Error_PodeCausarUmErro

    ‘ Inclua aqui código que possa gerar um erro
    ...
Exit_PodeCausarUmErro:
    Exit Function
Erro_PodeCausarUmErro:
    ‘ Inclua aqui código para tratamento de erros.
    ...
    ‘ Continua execução na rotina de saída para sair da função
    Resume Exit_PodeCausarUmErro
End Function
```

Tratando os erros em procedimento aninhados

Quando ocorre um erro em um procedimento aninhado que não tem uma rotina de tratamento de erros ativada, o Visual Basic procura retroativamente na lista de chamadas até encontrar uma rotina de tratamento de erros em um outro procedimento, em lugar de simplesmente parar a execução. Isso dá ao seu código a oportunidade de corrigir o erro dentro de um outro procedimento. Por exemplo, suponha que o procedimento A chame o procedimento B e que o procedimento B chame o procedimento C. Se ocorrer um erro no procedimento C e não houver uma rotina de tratamento de erros ativada, o Visual Basic verifica se existe no procedimento B e depois no procedimento A, uma rotina de tratamento de erros ativada. Se existir, a execução passa para essa rotina de tratamento de erros. Se não existir, a execução para e é exibida uma mensagem de erro.

O Visual Basic também procura retroativamente na lista de chamadas por uma rotina de tratamento de erros ativada, quando ocorre um erro dentro de uma rotina de tratamento de erros ativa. Pode-se forçar o Visual Basic a fazer essa procura, gerando um erro dentro de uma rotina de tratamento de erros ativa com o método *Raise* do objeto *Err*.. Se ocorrer um erro não previsto e você gerar novamente esse erro dentro da rotina de tratamento de erros, a execução passará retroativamente pela lista de chamadas para localizar uma outra rotina de tratamento de erros, que pode estar configurada para manipular o erro.

Por exemplo, suponha que o procedimento C tem uma rotina de tratamento de erros ativada, mas essa rotina não corrige o erro que ocorreu. Após verificar todos os erros que você previu, a rotina de tratamento de erros poderá gerar novamente o erro original. A execução então volta pela lista de chamadas para a rotina de tratamento de erros do procedimento B, caso exista, dando oportunidade para que essa rotina de tratamento de erros corrija o erro. Se não existir uma rotina de tratamento de erros no procedimento B, ou se ela não corrigir o erro e o gerar novamente, a execução passará à rotina de tratamento de erros do Procedimento A, supondo que exista uma.

Para ilustrar esse conceito de uma outra maneira, suponha que você tem um procedimento aninhado que inclui tratamento de erro para um erro de incompatibilidade de tipo, que você previu. Em algum ponto, ocorre um erro de divisão por zero, que você não previu, dentro do procedimento C. Se você tiver incluído uma instrução para gerar novamente o erro original, a execução voltará pela lista de chamadas para uma outra rotina de tratamento de erros ativada, caso exista. Se você corrigiu um erro de divisão por zero em outro procedimento da lista de chamadas, o erro será corrigido. Se o seu código não repetir o erro, então o procedimento continuará a ser executado,

sem corrigir o erro de divisão por zero. Isso, por sua vez, poderá causar outros erros dentro do conjunto de procedimentos aninhados.

Em resumo, o Visual Basic procura na lista de chamadas por uma rotina de tratamento de erros ativada se:

- Ocorrer um erro em um procedimento que não inclui uma rotina de tratamento de erros ativada.
- Ocorrer um erro dentro de uma rotina de tratamento de erros ativa. Se você utilizar o método *Raise* do objeto *Err* para gerar um erro, você poderá forçar o Visual Basic a procurar retroativamente na lista de chamadas por uma rotina de tratamento de erros ativada.

Obtendo informações sobre um erro

Quando a execução passa para a rotina de tratamento de erros, o seu código precisa determinar o erro ocorrido e tratá-lo. O Visual Basic e o Microsoft Excel oferecem vários elementos de linguagem que pode-se utilizar para obter informações sobre um erro específico. Cada um serve para diferentes tipos de erros. Como os erros podem ocorrer em diferentes partes do seu aplicativo, você precisará determinar qual utilizar no seu código, com base nos erros que você espera.

Os elementos de linguagem disponíveis para tratamento de erros incluem:

- objeto Err
- objeto Error e a coleção Errors
- método AccessError
- evento Error

O objeto Err

Este objeto é oferecido pelo Visual Basic. Quando ocorre um erro do Visual Basic, as informações sobre esse erro são armazenadas no objeto *Err*. Este objeto mantêm informações sobre um único erro de cada vez. Quando ocorre um novo erro, o objeto *Err* é atualizado de forma a conter informações sobre ele.

Para obter informações sobre um determinado erro, você ode utilizar as propriedades e métodos do objeto Err. A propriedade *Number* é a propriedade padrão; ela retorna o número de identificação do erro que ocorreu. A propriedade *Description* do objeto *Err* retorna a seqüência descritiva associada a um erro. O método *Clear* limpa as atuais informações de erro de objeto *Err*, enquanto o método *Raise* gera um erro específico e preenche as propriedades do objeto r com as informações sobre esse erro.

O exemplo a seguir mostra como utilizar o objeto *Err* em um procedimento que pode causar um erro de incompatibilidade de tipo:

```
Function PodeCausarUmErro()
    ‘ Declara constante que representa o erro provável
    Const conTipoIncompativel as Integer = 13

    ‘ Ativa rotina de tratamento de erros
    On Error GoTo Error_PodeCausarUmErro

    ‘ Inclua aqui código que possa gerar um erro
    ...
Exit_PodeCausarUmErro:
```

```
    Exit Function
Erro_PodeCausarUmErro:
    If Err = comTipoIncompativel Then
        ‘ Inclua aqui código para tratamento de erros.
        ...
    Else
        ‘ Gera novamente o erro original
        Dim intErrNum as Integer
        IntErrNum = Err
        Err.Clear
        Err.Raise intErrNum
    End If
    ‘ Continua execução na rotina de saída para sair da função
    Resume Exit_PodeCausarUmErro
End Function
```

Observe que no exemplo precedente o método *Raise* é utilizado para repetir o erro original. Se ocorrer um erro que não o de incompatibilidade de tipo a execução passará pela lista de chamadas para uma outra rotina de tratamento de erros ativa, caso exista.

O objeto *Err* lhe fornece todas as informações necessárias a respeito de erros do Visual Basic. Entretanto ele não dá informações completas sobre erros do Microsoft Excel ou do mecanismo de banco de dados Microsoft Jet. O Microsoft Excel e os objetos de acesso a dados (DAO) fornecem elementos de linguagem adicionais para ajudá-lo com estes erros.

O objeto Error e a coleção Errors

O objeto *Error* e a coleção *Erros* são fornecidos pelo DAO. O objeto *Error* representa um erro DAO. Uma única operação DAO pode causar vários erros, especialmente se você estiver efetuando operações ODBC. Cada erro que ocorre durante uma determinada operação de acesso a dados tem um objeto Error associado. Todos os objetos Erros associados a uma determinada operação DAO são armazenados na coleção *Errors*, sendo o erro de nível inferior o primeiro objeto na coleção e o erro de nível superior o último objeto na coleção.

Quando ocorre um erro DAO, o objeto *Err* do Visual Basic contém o número de erro do primeiro objeto da coleção *Errors*. Para determinar se ocorreram outros erros DAO, verifique a coleção *Errors*. Os valores das propriedades *Number* e *Description* do primeiro objeto *Error* da coleção *Errors* devem coincidir com os valores das propriedades Number e *Description* do objeto *Err* do Visual Basic.

O método AccessError

Pode-se utilizar o método do objeto *Err* para gerar um erro do Visual Basic que de fato não ocorreu e determinar a seqüência descritiva associada aquele erro. Entretanto, você não pode utilizar o método *Raise* para gerar um erro do Microsoft Excel ou um erro DAO. Para determinar a seqüência descritiva associada a um erro do Microsoft Excel ou a um erro DAO que de fato não ocorreu, utilize o método *AccessError*.

O evento Error

Pode-se utilizar o evento *Error* para interceptar erros que ocorrem em um formulário ou relatório do Microsoft Excel. Por exemplo, se um usuário tentar digitar texto em um campo cujo tipo de dados é Data/Hora, ocorrerá o evento *Error*. Se você adicionar um procedimento de evento *Error* a um formulário Funcionários, por exemplo, e tentar inserir um valor de texto no campo DataDeContratação, o procedimento de evento *Error* será executado.

O procedimento de evento *Error* ocupa um argumento 'inteiro', o *DataErr*. Quando um procedimento de evento *Error* é executado, o argumento *DataErr* contém o número do erro do Microsoft Excel que ocorreu. Verificar o valor do argumento *DataErr* dentro do procedimento de evento é a única maneira de se determinar o número que ocorreu. O objeto *Err* não é preenchido com informações quando o evento Error ocorre. Pode-se utilizar o valor do argumento *DataErr* juntamente com o método *AccessError* para determinar o número do erro e sua seqüência descritiva.

Observação: a instrução *Error* e a função *Error* são oferecidos somente para fins de compatibilidade retroativa. Ao escrever um novo código, utilize os objetos *Err* e *Error*, a função *AccessError* e o evento *Error* para obter informações sobre um erro.

Procedimentos automáticos

Um procedimento automático entra em execução automaticamente sempre que ocorrer um evento de um conjunto específico de eventos. Podemos associar procedimentos automáticos com eventos em nível de pasta de trabalho ou em nível de planilha.

No primeiro caso, os eventos automáticos para pasta de trabalho são:

Auto_Open: ocorre assim que o usuário abre a pasta

Auto_Close: ocorre assim que a pasta fechar (logo antes da ação se completar)

Para usar estes eventos automáticos você deverá criar um procedimento e nomeá-lo com estes nomes, como no exemplo abaixo:

```
Sub Auto_Open()
    Worksheets(1).Activate
    Range("A1").Select
End Sub
```

No segundo caso, quando desejamos trabalhar com eventos automáticos em nível de planilha, usamos os eventos:

Auto_Activate: quando desejamos executar algum código no instante em que uma planilha for ativada.

Auto_Deactivate: idem quando ela for desativada.

Para saber mais

Internet

OfficeVBA (www.officevba.com): Ótimo site sobre desenvolvimento em VBA, incluindo uma série de arquivos técnicos e tutoriais, além de forum de discussão com outros desenvolvedores.

The Spreadsheet Page (www.j-walk.com/ss/index.html): Tudo sobre as mais diversas planilhas e com seção especial sobre o Microsoft Excel. Possui uma boa seção de links para diversos outros sites sobre o assunto.

Office Power (www.office-power.com/index.htm): Bom site sobre os aplicativos do Microsoft Office, contendo arquivos técnicos sobre o Excel e VBA

Microsoft Office Developer Web Forum (http://www.microsoft.com/exceldev/e-a&sa.htm): Site oficial da Microsoft para desenvolvedores na plataforma Office.

Microsoft Support Knowledge Base (http://search.support.microsoft.com/kb/c.asp): O tira-dúvidas da Microsoft. Um grande banco de dados de informações sobre todos os produtos da empresa.

ASAP utilities (www.asap-utilities.com): Oferece um conjunto de utilitários que agrega uma série de novos comandos ao Excel

Livros

Microsoft Office 97 Visual Basic Programmer's Guide: referência sobre programação e customização dos aplicativos do Microsoft Office. Existe uma versão disponível online em http://www.microsoft.com/officedev/articles/Opg/toc/pgtoc.htm