VisuAlg - Editor e Interpretador de Pseudocódigos

**Objetivos**

A idéia de escrever este programa nasceu a partir de uma necessidade: uma ferramenta que permitisse aos alunos iniciantes em programação o exercício dos seus conhecimentos num ambiente próximo da realidade. Em minha experiência como professor desta disciplina, tenho notado que a abstração de "rodar o chinês", ou seja, de executar um programa apenas no papel, é um grande obstáculo (quase intransponível para alguns) no aprendizado das técnicas de elaboração de algoritmos. Por outro lado, submeter um iniciante aos rigores de uma linguagem de programação como Pascal ou ao "esoterismo" do C também me parecia exagerado. O ideal seria uma linguagem mais simples, parecida com o "Portugol", de grande popularidade nos meios acadêmicos e presente nos livros mais utilizados; com ela, os princípios básicos da programação estruturada poderiam ser ensinados sem que a curva de aprendizagem fosse íngreme. Além disso, esta ferramenta deveria também ser capaz de simular o que acontece na tela do computador com o uso dos famosos comandos "leia" e "escreva", bem como possibilitar a verificação dos valores das variáveis, o acompanhamento passo a passo da execução de um algoritmo (pelo seu grande valor didático), e até mesmo suportar um modo simples de depuração. Aliado a tudo isto, deveria estar um editor de texto com recursos razoáveis (tais como abrir e salvar arquivos) e que dispusesse de todos os principais recursos de um ambiente gráfico.

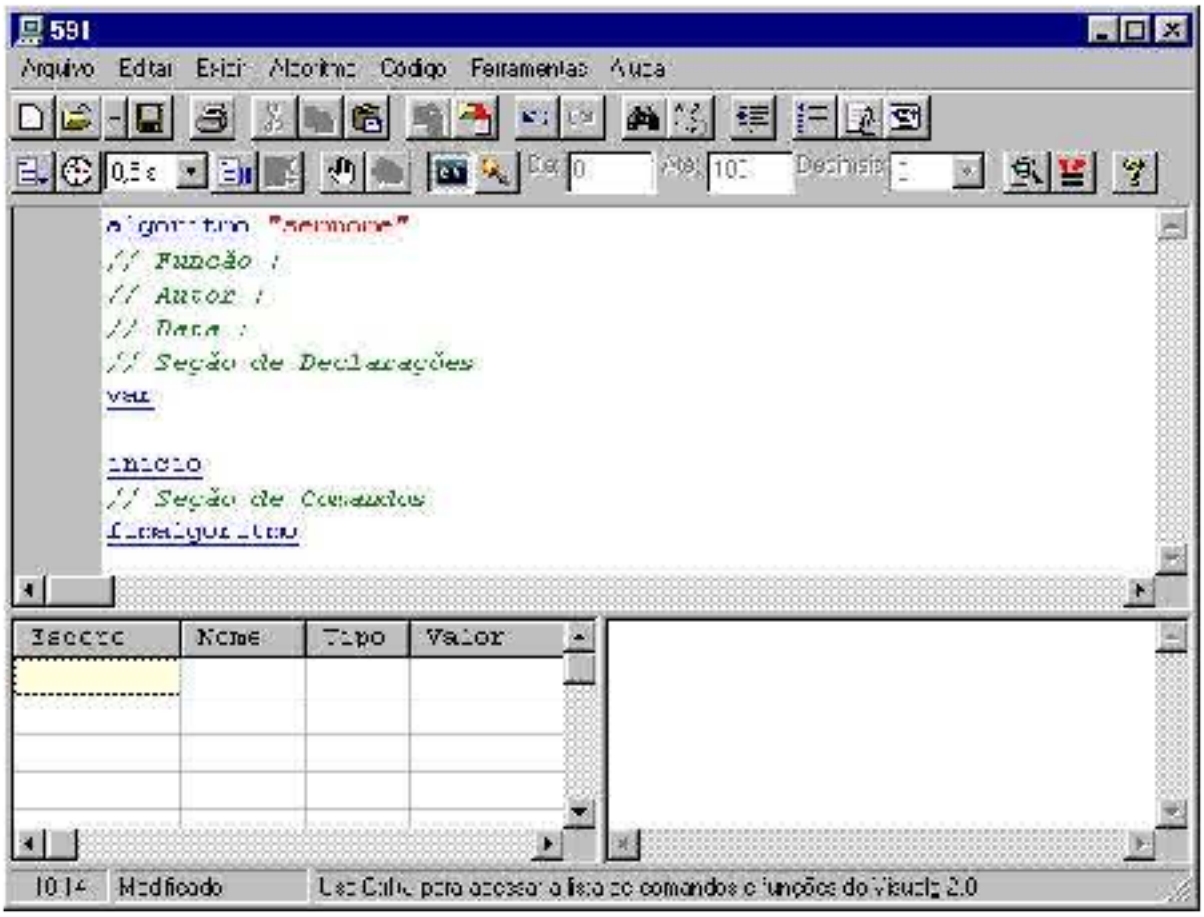
O VisuAlg é para mim a concretização desta idéia. Espero que, colocando-o em domínio público (numa versão *freeware*), possa ajudar professores e alunos de programação, e também ser ajudado por eles através de sugestões e críticas que visem sempre a sua melhoria. A idéia básica é manter o VisuAlg simples: deve ser como as rodinhas de apoio que uma criança usa ao aprender a andar de bicicleta, e que são retiradas quando deixam de ser necessárias. Isto não quer dizer que o VisuAlg não possa ou deva ser melhorado: conto com a colaboração de todos que vierem a utilizá-lo.

**Instalação e Requerimentos de Hardware**

O VisuAlg é um programa simples, que não depende de DLLs, OCXs ou outros componentes. Sua instalação não copia arquivos para nenhuma outra pasta a não ser aquela em que for instalado, e exige cerca de 1 MB de espaço em disco. Pode ser executado sob Windows 95 ou posterior, e tem melhor aparência com resolução de vídeo de 800x600 ou maior.

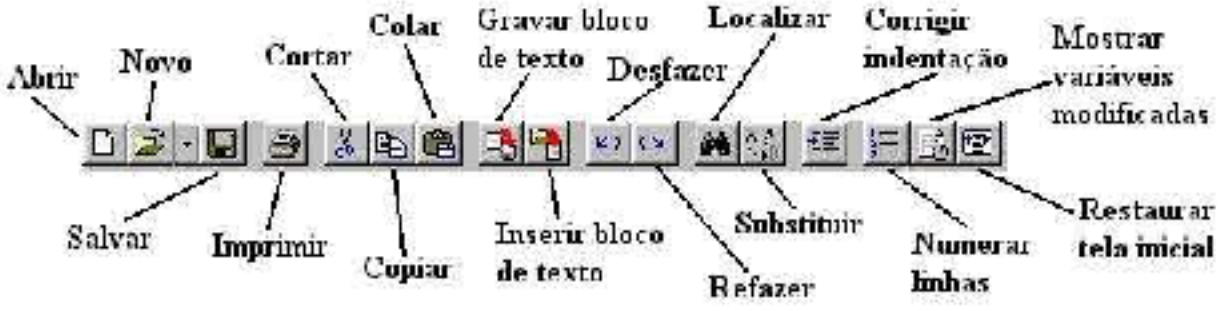
**A Tela Principal do VisuAlg**

A tela do VisuAlg compõe-se da barra de tarefas, do editor de textos (que toma toda a sua metade superior), do quadro de variáveis (no lado esquerdo da metade inferior), do simulador de saída (no correspondente lado direito) e da barra de *status*. Quando o programa é carregado, já apresenta no editor um "esqueleto" de pseudocódigo, com a intenção de poupar trabalho ao usuário e de mostrar o formato básico que deve ser seguido. Explicaremos a seguir cada componente da interface do VisuAlg.




**A Barra de Tarefas**

Contém os comandos mais utilizados no VisuAlg (estes comandos também podem ser acessados pelo menu ou por atalhos no teclado).



**Abrir (Ctrl-A):** Abre um arquivo anteriormente gravado, substituindo o texto presente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Novo (Ctrl-N):** Cria um novo "esqueleto" de pseudocódigo, substituindo o texto presente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Salvar (Ctrl-S):** Grava imediatamente o texto presente no editor. Na primeira vez que um novo texto é gravado, o VisuAlg pede seu nome e localização.
**Imprimir:** Imprime imediatamente na impressora padrão o texto presente no editor. Para configurar a impressão, use o comando Imprimir do menu Arquivo (acessível também pelo atalho *Ctrl-P*).
**Cortar (Ctrl-X):** Apaga texto selecionado, armazenando-o em uma área de transferência.
**Copiar (Ctrl-C):** Copia o texto selecionado para a área de transferência.
**Colar (Ctrl-V):** Copia texto da área de transferência para o local em que está o cursor.
**Gravar bloco de texto:** Permite a gravação em arquivo de um texto selecionado no editor. A extensão sugerida para o nome do arquivo é *.inc*.
**Inserir bloco de texto:** Permite a inserção do conteúdo de um arquivo. A extensão sugerida para o nome do arquivo é *.inc*.
**Desfazer (Ctrl-Z):** Desfaz último comando efetuado.
**Refazer (Shift-Ctrl-Z):** Refaz último comando desfeito.
**Localizar (Ctrl-L):** Localiza no texto presente no editor determinada palavra especificada.
**Substituir (Ctrl-U):** Localiza no texto presente no editor determinada palavra especificada, substituindo-a por outra.
**Corrigir Indentação (Ctrl-G)**: Corrige automaticamente a *indentação* (ou tabulação) do pseudocódigo, tabulando cada comando interno com espaços à esquerda.
**Numerar linhas:** Ativa ou desativa a exibição dos números das linhas na área à esquerda do editor. A linha e a coluna do editor em que o cursor está em um determinado momento também são mostradas na barra de *status* (parte inferior da tela). Por motivos técnicos, esta opção é automaticamente desativada durante a execução do pseudocódigo, mas volta a ser ativada logo em seguida.
**Mostrar variáveis modificadas**: Ativa ou desativa a exibição da variável que está sendo modificada. Como o número de variáveis pode ser grande, muitas podem estar fora da janela de visualização; quando esta característica está ativada, o VisuAlg rola a grade de exibição de modo que cada variável fique visível no momento em está sendo modificada. Este recurso é especialmente útil quando se executa um pseudocódigo passo a passo. Por questões de desempenho, a configuração padrão desta característica é *desativada*, quando o pseudocódigo está sendo executado automaticamente. No entanto, basta clicar este botão para executá-lo automaticamente com a exibição ativada. No final da execução, a configuração volta a ser *desativada*.
**Restaurar tela inicial:** Retorna a divisão da tela ao formato inicial, caso você tenha mudado o tamanho da área do editor de texto, quadro de variáveis ou simulador de saída.


**Executar (F9):** Inicia (ou continua) a execução automática do pseudocódigo.
**Executar com *timer* (Shift-F9):**: Insere um atraso (que pode ser especificado no intervalo ao lado) antes da execução de cada linha. Também realça em fundo azul o comando que está sendo executado, da mesma forma que na execução passo a passo.
**Intervalo do *timer*:** Atraso em cada linha, para quando se deseja executar o pseudocódigo com *timer*.
**Passo (F8):** Inicia (ou continua) a execução linha por linha do pseudocódigo, dando ao usuário a oportunidade de acompanhar o fluxo de execução, os valores das variáveis e a pilha de ativação dos subprogramas.
**Parar (Ctrl-F2):** Termina imediatamente a execução do pseudocódigo. Evidentemente, este botão fica desabilitado quando o pseudocódigo não está sendo executado.
**Liga/desliga *breakpoint* (F5):** Insere/remove um ponto de parada na linha em que esteja o cursor. Estes pontos de parada são úteis para a depuração e acompanhamento da execução dos pseudocódigos, pois

permitem a verificação dos valores das variáveis e da pilha de ativação de subprogramas.
**Desmarcar todos os *breakpoints* (Ctrl-F5):** Desativa todos os *breakpoints* que estejam ativados naquele momento.
**Executar em modo DOS:** Com esta opção ativada, tanto a entrada como a saída-padrão passa a ser uma janela que imita o DOS, simulando a execução de um programa neste ambiente.
**Gerar valores aleatórios:** Ativa a geração de valores aleatórios que substituem a digitação de dados. A faixa padrão de valores gerados é de 0 a 100 inclusive, mas pode ser modificada (basta alterar intervalo ao lado). Para a geração de dados do tipo caractere, não há uma faixa pré-estabelecida: os dados gerados serão sempre *strings* de 5 letras maiúsculas.
**Intervalo dos valores aleatórios:** Faixa de valores que serão gerados automaticamente, quando esta opção estiver ativada.
**Perfil (F7):** Após a execução de um pseudocódigo, exibe o número de vezes que cada umas das suas linhas foi executada. É útil para a análise de eficiência (por exemplo, nos métodos de ordenação).
**Mostrar pilha de ativação (Ctrl-F3):** Exibe a pilha de subprogramas ativados num dado momento. Convém utilizar este comando em conjunto com *breakpoints* ou com a execução passo a passo.
**Ajuda (F1):** Possibilita acesso às páginas de ajuda e às informações sobre o VisuAlg.

**Quadro de Variáveis**

É formado por uma grade na qual são mostrados o escopo de cada variável (se for do programa principal, será global; se for local, será apresentado o nome do subprograma onde foi declarada), seus nomes (também com os índices, nos casos em que sejam vetores), seu tipo ("I" para inteiro, "R" para real, "C" para caractere e "L" para lógico) e o seu valor corrente. A versão atual do VisuAlg permite a visualização de até 500 variáveis (contando individualmente cada elemento dos vetores).

**A Barra de *Status***

Situada na parte inferior da tela, esta barra contém dois painéis: o primeiro mostra a linha e a coluna onde o cursor está, e o segundo mostra a palavra *Modificado* no caso em que o pseudocódigo tenha sido alterado desde que foi carregado ou salvo pela última vez. Nesta barra, há ainda um terceiro painel disponível, que ainda não tem um uso específico na atual versão.

Menu do VisuAlg

Este menu compõe-se de 7 partes:

**Arquivo:** Possui os comandos para se abrir, salvar e imprimir algoritmos:

**Novo:** Cria um novo "esqueleto" de pseudocódigo, substituindo o texto existente no editor. Se este texto anterior tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Abrir:** Abre o texto de um pseudocódigo anteriormente gravado, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Salvar:** Salva imediatamente o texto presente no editor. Caso seja a primeira vez que um novo texto é gravado, o VisuAlg pedirá o nome do arquivo e sua localização.
**Salvar como:** Permite salvar o texto presente no editor exibindo antes uma janela na qual se pode escolher o nome do arquivo e sua localização.
**Enviar por email:** Permite o envio por email do texto presente no editor.
**Imprimir:** Permite a impressão do algoritmo corrente, mostrando antes a janela de configuração de impressão (o correspondente botão da barra de tarefas imprime imediatamente o texto do pseudocódigo na impressora padrão).
**Sair:** Abandona o VisuAlg.

Além destes comandos, há ainda a lista dos 5 últimos algoritmos utilizados, que podem ser abertos diretamente ao se escolher o seu nome.

**Editar:** Além dos conhecidos comandos de um editor de texto (copiar, cortar, colar, desfazer, refazer, selecionar tudo, localizar, localizar de novo, substituir), há também as seguintes opções:

**Corrigir indentação:** Corrige automaticamente a *indentação* do pseudocódigo, tabulando cada comando interno com espaços à esquerda.
**Gravar bloco de texto:** Permite a gravação em arquivo de um texto selecionado no editor. A extensão sugerida para o nome do arquivo é *.inc*.
**Inserir bloco de texto:** Permite a inserção do conteúdo de um arquivo. A extensão sugerida para o nome do arquivo é *.inc*.

**Exibir:** Possui os comandos para ativar/desativar as seguintes características:

**Número de linhas:** Ativa/desativa a exibição da numeração das linhas na área à esquerda do editor. A numeração corrente da posição do cursor também é mostrada na primeira parte da barra de status, situada na parte inferior da tela. Por motivos técnicas, a numeração é desativada durante a execução do pseudocódigo, voltando à situação anterior logo em seguida.
**Variáveis modificadas:** Ativa/desativa a exibição da variável que está sendo modificada. Como o número de variáveis pode ser grande, muitas podem estar fora da janela de visualização; quando esta característica está ativada, o VisuAlg rola a grade de exibição de modo que cada variável fique visível no momento em está sendo modificada. Este recurso é especialmente útil quando se executa um pseudocódigo passo a passo. Por questões de desempenho, a configuração padrão desta característica é *desativada*, quando o pseudocódigo está sendo executado automaticamente. No entanto, basta clicar este botão para executá-lo automaticamente com a exibição ativada. No final da execução, a configuração volta a ser *desativada*.

**Pseudocódigo:** Contém os comandos relativos à execução do algoritmo:

**Executar:** Inicia (ou continua) a execução automática do pseudocódigo.
**Passo a passo:** Inicia (ou continua) a execução linha por linha do pseudocódigo, dando ao usuário a oportunidade de acompanhar o fluxo de execução, os valores das variáveis e a pilha de ativação dos subprogramas.
**Executar com *timer*:** Insere um atraso (que pode ser especificado) antes da execução de cada linha. Também realça em fundo azul o comando que está sendo executado, da mesma forma que na execução passo a passo.
**Parar:** Termina imediatamente a execução do pseudocódigo. Evidentemente, este item fica desabilitado quando o pseudocódigo não está sendo executado.
**Liga/desliga *breakpoint*:** Insere/remove um ponto de parada na linha em que esteja o cursor. Estes pontos

de parada são úteis para a depuração e acompanhamento da execução dos pseudocódigos, pois permitem a verificação dos valores das variáveis e da pilha de ativação de subprogramas.
**Desmarcar todos os *breakpoints*:** Desativa todos os *breakpoints* que estejam ativados naquele momento.
**Executar em modo DOS:** Com esta opção ativada, tanto a entrada como a saída-padrão passa a ser uma janela que imita o DOS, simulando a execução de um programa neste ambiente.
**Gerar valores aleatórios:** Ativa a geração de valores aleatórios que substituem a digitação de dados. A faixa padrão de valores gerados é de 0 a 100 inclusive, mas pode ser modificada. Para a geração de dados do tipo caractere, não há uma faixa pré-estabelecida: os dados gerados serão sempre *strings* de 5 letras maiúsculas.
**Perfil:** Após a execução de um pseudocódigo, exibe o número de vezes que cada umas das suas linhas foi executada. É útil para a análise de eficiência (por exemplo, nos métodos de ordenação).
**Pilha de ativação:** Exibe a pilha de subprogramas ativados num dado momento. Convém utilizar este comando em conjunto com *breakpoints* ou com a execução passo a passo.

**Linguagens:** Permite a tradução automático do pseudocódigo presente no editor para outras linguagens de programação. Atualmente, apenas a tradução para Pascal está implementada, mas ainda em fase de testes.

**Ferramentas:** Neste menu, é possível configurar algumas opções do VisuAlg: cores e tipos de letras na exibição do pseudocódigo, número de espaços para indentação automática, etc.

**Ajuda:** Entre outras coisas, possibilita acesso às páginas de ajuda e às informações sobre o VisuAlg.

Menu do VisuAlg

Este menu compõe-se de 7 partes:

**Arquivo:** Possui os comandos para se abrir, salvar e imprimir algoritmos:

**Novo:** Cria um novo "esqueleto" de pseudocódigo, substituindo o texto existente no editor. Se este texto anterior tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Abrir:** Abre o texto de um pseudocódigo anteriormente gravado, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Salvar:** Salva imediatamente o texto presente no editor. Caso seja a primeira vez que um novo texto é gravado, o VisuAlg pedirá o nome do arquivo e sua localização.
**Salvar como:** Permite salvar o texto presente no editor exibindo antes uma janela na qual se pode escolher o nome do arquivo e sua localização.
**Enviar por email:** Permite o envio por email do texto presente no editor.
**Imprimir:** Permite a impressão do algoritmo corrente, mostrando antes a janela de configuração de impressão (o correspondente botão da barra de tarefas imprime imediatamente o texto do pseudocódigo na impressora padrão).
**Sair:** Abandona o VisuAlg.

Além destes comandos, há ainda a lista dos 5 últimos algoritmos utilizados, que podem ser abertos diretamente ao se escolher o seu nome.

**Editar:** Além dos conhecidos comandos de um editor de texto (copiar, cortar, colar, desfazer, refazer, selecionar tudo, localizar, localizar de novo, substituir), há também as seguintes opções:

**Corrigir indentação:** Corrige automaticamente a *indentação* do pseudocódigo, tabulando cada comando interno com espaços à esquerda.
**Gravar bloco de texto:** Permite a gravação em arquivo de um texto selecionado no editor. A extensão sugerida para o nome do arquivo é *.inc*.
**Inserir bloco de texto:** Permite a inserção do conteúdo de um arquivo. A extensão sugerida para o nome do arquivo é *.inc*.

**Exibir:** Possui os comandos para ativar/desativar as seguintes características:

**Número de linhas:** Ativa/desativa a exibição da numeração das linhas na área à esquerda do editor. A numeração corrente da posição do cursor também é mostrada na primeira parte da barra de status, situada na parte inferior da tela. Por motivos técnicas, a numeração é desativada durante a execução do pseudocódigo, voltando à situação anterior logo em seguida.
**Variáveis modificadas:** Ativa/desativa a exibição da variável que está sendo modificada. Como o número de variáveis pode ser grande, muitas podem estar fora da janela de visualização; quando esta característica está ativada, o VisuAlg rola a grade de exibição de modo que cada variável fique visível no momento em está sendo modificada. Este recurso é especialmente útil quando se executa um pseudocódigo passo a passo. Por questões de desempenho, a configuração padrão desta característica é *desativada*, quando o pseudocódigo está sendo executado automaticamente. No entanto, basta clicar este botão para executá-lo automaticamente com a exibição ativada. No final da execução, a configuração volta a ser *desativada*.

**Pseudocódigo:** Contém os comandos relativos à execução do algoritmo:

**Executar:** Inicia (ou continua) a execução automática do pseudocódigo.
**Passo a passo:** Inicia (ou continua) a execução linha por linha do pseudocódigo, dando ao usuário a oportunidade de acompanhar o fluxo de execução, os valores das variáveis e a pilha de ativação dos subprogramas.
**Executar com *timer*:** Insere um atraso (que pode ser especificado) antes da execução de cada linha. Também realça em fundo azul o comando que está sendo executado, da mesma forma que na execução passo a passo.
**Parar:** Termina imediatamente a execução do pseudocódigo. Evidentemente, este item fica desabilitado quando o pseudocódigo não está sendo executado.
**Liga/desliga *breakpoint*:** Insere/remove um ponto de parada na linha em que esteja o cursor. Estes pontos de parada são úteis para a depuração e acompanhamento da execução dos pseudocódigos, pois permitem a verificação dos valores das variáveis e da pilha de ativação de subprogramas.

**Desmarcar todos os *breakpoints*:** Desativa todos os *breakpoints* que estejam ativados naquele momento.
**Executar em modo DOS:** Com esta opção ativada, tanto a entrada como a saída-padrão passa a ser uma janela que imita o DOS, simulando a execução de um programa neste ambiente.
**Gerar valores aleatórios:** Ativa a geração de valores aleatórios que substituem a digitação de dados. A faixa padrão de valores gerados é de 0 a 100 inclusive, mas pode ser modificada. Para a geração de dados do tipo caractere, não há uma faixa pré-estabelecida: os dados gerados serão sempre *strings* de 5 letras maiúsculas.
**Perfil:** Após a execução de um pseudocódigo, exibe o número de vezes que cada umas das suas linhas foi executada. É útil para a análise de eficiência (por exemplo, nos métodos de ordenação).
**Pilha de ativação:** Exibe a pilha de subprogramas ativados num dado momento. Convém utilizar este comando em conjunto com *breakpoints* ou com a execução passo a passo.

**Linguagens:** Permite a tradução automático do pseudocódigo presente no editor para outras linguagens de programação. Atualmente, apenas a tradução para Pascal está implementada, mas ainda em fase de testes.

**Ferramentas:** Neste menu, é possível configurar algumas opções do VisuAlg: cores e tipos de letras na exibição do pseudocódigo, número de espaços para indentação automática, etc.

**Ajuda:** Entre outras coisas, possibilita acesso às páginas de ajuda e às informações sobre o VisuAlg.

Menu do VisuAlg

Este menu compõe-se de 7 partes:

**Arquivo:** Possui os comandos para se abrir, salvar e imprimir algoritmos:

**Novo:** Cria um novo "esqueleto" de pseudocódigo, substituindo o texto existente no editor. Se este texto anterior tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Abrir:** Abre o texto de um pseudocódigo anteriormente gravado, substituindo o texto existente no editor. Se este tiver sido modificado, o VisuAlg pedirá sua confirmação para salvá-lo antes que seja sobreposto.
**Salvar:** Salva imediatamente o texto presente no editor. Caso seja a primeira vez que um novo texto é gravado, o VisuAlg pedirá o nome do arquivo e sua localização.
**Salvar como:** Permite salvar o texto presente no editor exibindo antes uma janela na qual se pode escolher o nome do arquivo e sua localização.
**Enviar por email:** Permite o envio por email do texto presente no editor.
**Imprimir:** Permite a impressão do algoritmo corrente, mostrando antes a janela de configuração de impressão (o correspondente botão da barra de tarefas imprime imediatamente o texto do pseudocódigo na impressora padrão).
**Sair:** Abandona o VisuAlg.

Além destes comandos, há ainda a lista dos 5 últimos algoritmos utilizados, que podem ser abertos diretamente ao se escolher o seu nome.

**Editar:** Além dos conhecidos comandos de um editor de texto (copiar, cortar, colar, desfazer, refazer, selecionar tudo, localizar, localizar de novo, substituir), há também as seguintes opções:

**Corrigir indentação:** Corrige automaticamente a *indentação* do pseudocódigo, tabulando cada comando interno com espaços à esquerda.
**Gravar bloco de texto:** Permite a gravação em arquivo de um texto selecionado no editor. A extensão sugerida para o nome do arquivo é *.inc*.
**Inserir bloco de texto:** Permite a inserção do conteúdo de um arquivo. A extensão sugerida para o nome do arquivo é *.inc*.

**Exibir:** Possui os comandos para ativar/desativar as seguintes características:

**Número de linhas:** Ativa/desativa a exibição da numeração das linhas na área à esquerda do editor. A numeração corrente da posição do cursor também é mostrada na primeira parte da barra de status, situada na parte inferior da tela. Por motivos técnicas, a numeração é desativada durante a execução do pseudocódigo, voltando à situação anterior logo em seguida.
**Variáveis modificadas:** Ativa/desativa a exibição da variável que está sendo modificada. Como o número de variáveis pode ser grande, muitas podem estar fora da janela de visualização; quando esta característica está ativada, o VisuAlg rola a grade de exibição de modo que cada variável fique visível no momento em está sendo modificada. Este recurso é especialmente útil quando se executa um pseudocódigo passo a passo. Por questões de desempenho, a configuração padrão desta característica é *desativada*, quando o pseudocódigo está sendo executado automaticamente. No entanto, basta clicar este botão para executá-lo automaticamente com a exibição ativada. No final da execução, a configuração volta a ser *desativada*.

**Pseudocódigo:** Contém os comandos relativos à execução do algoritmo:

**Executar:** Inicia (ou continua) a execução automática do pseudocódigo.
**Passo a passo:** Inicia (ou continua) a execução linha por linha do pseudocódigo, dando ao usuário a oportunidade de acompanhar o fluxo de execução, os valores das variáveis e a pilha de ativação dos subprogramas.
**Executar com *timer*:** Insere um atraso (que pode ser especificado) antes da execução de cada linha. Também realça em fundo azul o comando que está sendo executado, da mesma forma que na execução passo a passo.
**Parar:** Termina imediatamente a execução do pseudocódigo. Evidentemente, este item fica desabilitado quando o pseudocódigo não está sendo executado.
**Liga/desliga *breakpoint*:** Insere/remove um ponto de parada na linha em que esteja o cursor. Estes pontos de parada são úteis para a depuração e acompanhamento da execução dos pseudocódigos, pois permitem a verificação dos valores das variáveis e da pilha de ativação de subprogramas.

**Desmarcar todos os *breakpoints*:** Desativa todos os *breakpoints* que estejam ativados naquele momento.
**Executar em modo DOS:** Com esta opção ativada, tanto a entrada como a saída-padrão passa a ser uma janela que imita o DOS, simulando a execução de um programa neste ambiente.
**Gerar valores aleatórios:** Ativa a geração de valores aleatórios que substituem a digitação de dados. A faixa padrão de valores gerados é de 0 a 100 inclusive, mas pode ser modificada. Para a geração de dados do tipo caractere, não há uma faixa pré-estabelecida: os dados gerados serão sempre *strings* de 5 letras maiúsculas.
**Perfil:** Após a execução de um pseudocódigo, exibe o número de vezes que cada umas das suas linhas foi executada. É útil para a análise de eficiência (por exemplo, nos métodos de ordenação).
**Pilha de ativação:** Exibe a pilha de subprogramas ativados num dado momento. Convém utilizar este comando em conjunto com *breakpoints* ou com a execução passo a passo.

**Linguagens:** Permite a tradução automático do pseudocódigo presente no editor para outras linguagens de programação. Atualmente, apenas a tradução para Pascal está implementada, mas ainda em fase de testes.

**Ferramentas:** Neste menu, é possível configurar algumas opções do VisuAlg: cores e tipos de letras na exibição do pseudocódigo, número de espaços para indentação automática, etc.

**Ajuda:** Entre outras coisas, possibilita acesso às páginas de ajuda e às informações sobre o VisuAlg.

A Linguagem de Programação do VisuAlg (3)

**Comandos de Repetição**

O VisuAlg implementa as três estruturas de repetição usuais nas linguagens de programação: o laço contado `para...ate...faca` (similar ao *for...to...do* do Pascal), e os laços condicionados `enquanto...faca` (similar ao *while...do*) e `repita...ate` (similar ao *repeat...until*). A sintaxe destes comandos é explicada a seguir.

**Para ... faça**

Esta estrutura repete uma seqüência de comandos um determinado número de vezes.

```
para <variável> de <valor-inicial> ate <valor-limite> [passo <incremento>] faca
   <seqüência-de-comandos>
fimpara
```

| | |
|---|---|
| `<variável >` | É a variável contadora que controla o número de repetições do laço. Na versão atual, deve ser necessariamente uma variável do tipo `inteiro`, como todas as expressões deste comando. |
| `<valor-inicial>` | É uma expressão que especifica o valor de inicialização da variável contadora antes da primeira repetição do laço. |
| `<valor-limite >` | É uma expressão que especifica o valor máximo que a variável contadora pode alcançar. |
| `<incremento >` | É opcional. Quando presente, precedida pela palavra `passo`, é uma expressão que especifica o incremento que será acrescentado à variável contadora em cada repetição do laço. Quando esta opção não é utilizada, o valor padrão de `<incremento>` é 1. Vale a pena ter em conta que também é possível especificar valores negativos para `<incremento>`. Por outro lado, se a avaliação da expressão `<incremento >` resultar em valor nulo, a execução do algoritmo será interrompida, com a impressão de uma mensagem de erro. |
| `fimpara` | Indica o fim da seqüência de comandos a serem repetidos. Cada vez que o programa chega neste ponto, é acrescentado à variável contadora o valor de `<incremento >`, e comparado a `<valor-limite >`. Se for menor ou igual (ou maior ou igual, quando `<incremento >` for negativo), a seqüência de comandos será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando que esteja após o `fimpara`. |

`<valor-inicial >`, `<valor-limite >` e `<incremento >` são avaliados uma **única vez** antes da execução da primeira repetição, e **não se alteram durante a execução do laço**, mesmo que variáveis eventualmente presentes nessas expressões tenham seus valores alterados.

No exemplo a seguir, os números de 1 a 10 são exibidos em ordem crescente.

```
algoritmo "Números de 1 a 10"
var j: inteiro
inicio
para j de 1 ate 10 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

**Importante:** Se, logo no início da primeira repetição, `<valor-inicial >` for maior que `<valor-limite >` (ou menor, quando `<incremento>` for negativo), o laço não será executado nenhuma vez. O exemplo a seguir não imprime nada.

```
algoritmo "Numeros de 10 a 1 (não funciona)"
var j: inteiro
inicio
para j de 10 ate 1 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

Este outro exempo, no entanto, funcionará por causa do **passo -1**:

```
algoritmo "Numeros de 10 a 1 (este funciona)"
var j: inteiro
inicio
para j de 10 ate 1 passo -1 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

**Enquanto ... faça**

Esta estrutura repete uma seqüência de comandos enquanto uma determinada condição (especificada através de uma expressão lógica) for satisfeita.

```
enquanto <expressão-lógica> faca
   <seqüência-de-comandos>
fimenquanto
```

| | |
|---|---|
| *<expressão-lógica>* | Esta expressão que é avaliada antes de cada repetição do laço. Quando seu resultado for VERDADEIRO, *<seqüência-de-comandos>* é executada. |
| `fimenquanto` | Indica o fim da *<seqüência-de-comandos>* que será repetida. Cada vez que a execução atinge este ponto, volta-se ao início do laço para que *<expressão-lógica>* seja avaliada novamente. Se o resultado desta avaliação for VERDADEIRO, a *<seqüência-de-comandos>* será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando após `fimenquanto`. |

O mesmo exemplo anterior pode ser resolvido com esta estrutura de repetição:

```
algoritmo "Números de 1 a 10 (com enquanto...faca)"
var j: inteiro
inicio
j <- 1
enquanto j <= 10 faca
   escreva (j:3)
   j <- j + 1
fimenquanto
fimalgoritmo
```

**Importante:** Como o laço `enquanto...faca` testa sua condição de parada **antes** de executar sua seqüência de comandos, esta seqüência poderá ser executada **zero ou mais vezes**.

**Repita ... até**

Esta estrutrura repete uma seqüência de comandos até que uma determinada condição (especificada através de uma expressão lógica) seja satisfeita.

```
repita
   <seqüência-de-comandos>
ate <expressão-lógica>
```

| | |
|---|---|
| `repita` | Indica o início do laço. |
| `ate <expressão-lógica>` | Indica o fim da *<seqüência-de-comandos>* a serem repetidos. Cada vez que o programa chega neste ponto, *<expressão-lógica>* é avaliada: se seu resultado for FALSO, os comandos presentes entre esta linha e a linha `repita` são executados; caso contrário, a execução prosseguirá a partir do primeiro comando após esta linha. |

Considerando ainda o mesmo exemplo:

```
algoritmo "Números de 1 a 10 (com repita)"
var j: inteiro
inicio
j <- 1
repita
   escreva (j:3)
   j <- j + 1
ate j > 10
fimalgoritmo
```

**Importante:** Como o laço `repita...ate` testa sua condição de parada **depois** de executar sua seqüência de comandos, esta seqüência poderá ser executada **uma ou mais vezes**.

**Comando Interrompa**

As três estruturas de repetição acima permitem o uso do comando `interrompa`, que causa uma saída imediata do laço. Embora esta técnica esteja de certa forma em desacordo com os princípios da programação estruturada, o comando `interrompa` foi incluído no VisuAlg por ser encontrado na literatura de introdução à programação e mesmo em linguagens como o Object Pascal (Delphi/Kylix), Clipper, VB, etc. Seu uso é exemplificado a seguir:

```
algoritmo "Números de 1 a 10 (com interrompa)"
var x: inteiro
inicio
x <- 0
repita
   x <- x + 1
   escreva (x:3)
   se x = 10 entao
      interrompa
   fimse
ate falso
fimalgoritmo
```

O VisuAlg permite ainda uma forma alternativa do comando `repita...ate`, com a seguinte sintaxe:

```
algoritmo "Números de 1 a 10 (com interrompa) II"
var x: inteiro
inicio
x <- 0
repita
   x <- x + 1
   escreva (x:3)
   se x = 10 entao
      interrompa
   fimse
fimrepita
fimalgoritmo
```

Com esta sintaxe alternativa, o uso do `interrompa` é obrigatório, pois é a única maneira de se sair do laço `repita...fimrepita`; caso contrário, este laço seria executado indeterminadamente.

A Linguagem de Programação do VisuAlg (4)

*Subprograma* é um programa que auxilia o programa principal através da realização de uma determinada subtarefa. Também costuma receber os nomes de *sub-rotina*, *procedimento*, *método* ou *módulo*. Os subprogramas são chamados dentro do corpo do programa principal como se fossem *comandos*. Após seu término, a execução continua a partir do ponto onde foi chamado. É importante compreender que a chamada de um subprograma simplesmente gera um **desvio provisório no fluxo de execução**.

Há um caso particular de subprograma que recebe o nome de *função*. Uma *função*, além de executar uma determinada tarefa, retorna um valor para quem a chamou, que é o resultado da sua execução. Por este motivo, a chamada de uma função aparece no corpo do programa principal como uma *expressão*, e não como um comando.

Cada subprograma, além de ter acesso às variáveis do programa que o chamou (são as variáveis *globais*), pode ter suas próprias variáveis (são as variáveis *locais*), que existem apenas durante sua chamada.

Ao se chamar um subprograma, também é possível passar-lhe determinadas informações que recebem o nome de *parâmetros* (são valores que, na linha de chamada, ficam entre os parênteses e que estão separados por vírgulas). A quantidade dos parâmetros, sua seqüência e respectivos tipos não podem mudar: devem estar de acordo com o que foi especificado na sua correspondente declaração.

Para se criar subprogramas, é preciso descrevê-los após a declaração das variáveis e antes do corpo do programa principal. O VisuAlg possibilita declaração e chamada de subprogramas nos moldes da linguagem Pascal, ou seja, procedimentos e funções com passagem de parâmetros por valor ou referência. Isso será explicado a seguir.

**Procedimentos**

Em VisuAlg, procedimento é um subprograma que não retorna nenhum valor (corresponde ao *procedure* do Pascal). Sua declaração, que deve estar entre o final da declaração de variáveis e a linha `inicio` do programa principal, segue a sintaxe abaixo:

```
procedimento <nome-de-procedimento> [(<seqüência-de-declarações-de-parâmetros>)]
// Seção de Declarações Internas
inicio
// Seção de Comandos
fimprocedimento
```

O *<nome-de-procedimento>* obedece as mesmas regras de nomenclatura das variáveis. Por outro lado, a *<seqüência-de-declarações-de-parâmetros>* é uma seqüência de

```
[var] <seqüência-de-parâmetros>: <tipo-de-dado>
```

separadas por ponto e vírgula. A presença (opcional) da palavra-chave `var` indica passagem de parâmetros por referência; caso contrário, a passagem será por valor.

Por sua vez, *<seqüência-de-parâmetros>* é uma seqüência de nomes de parâmetros (também obedecem a mesma regra de nomenclatura de variáveis) separados por vírgulas.

De modo análogo ao programa principal, a seção de declaração internas começa com a palavra-chave `var`, e continua com a seguinte sintaxe:

```
<lista-de-variáveis> : <tipo-de-dado>
```

Nos próximos exemplos, através de um subprograma `soma`, será calculada a soma entre os valores 4 e –9 (ou seja, será obtido o resultado 13) que o programa principal imprimirá em seguida. No primeiro caso, um **procedimento sem parâmetros** utiliza uma variável local `aux` para armazenar provisoriamente o resultado deste cálculo (evidentemente, esta variável é desnecessária, mas está aí apenas para ilustrar o exemplo), antes de atribuí-lo à variável global `res`:

```
procedimento soma
var aux: inteiro
inicio
// n, m e res são variáveis globais
aux <- n + m
res <- aux
fimprocedimento
```

No programa principal deve haver os seguintes comandos:

```
n <- 4
m <- -9
soma
escreva(res)
```

A mesma tarefa poderia ser executada através de um **procedimento com parâmetros**, como descrito abaixo:

```
procedimento soma (x,y: inteiro)
inicio
// res é variável global
res <- x + y
fimprocedimento
```

No programa principal deve haver os seguintes comandos:

```
n <- 4
m <- -9
soma(n,m)
escreva(res)
```

A passagem de parâmetros do exemplo acima chama-se **passagem por valor**. Neste caso, o subprograma simplesmente recebe um valor que utiliza durante sua execução. Durante essa execução, os parâmetros passados por valor são análogos às suas variáveis locais, mas com uma única diferença: receberam um valor inicial no momento em que o subprograma foi chamado.

**Funções**

Em VisuAlg, função é um subprograma que retorna um valor (corresponde ao *function* do Pascal). De modo análogo aos procedimentos, sua declaração deve estar entre o final da declaração de variáveis e a linha `inicio` do programa principal, e segue a sintaxe abaixo:

```
funcao <nome-de-função> [(<seqüência-de-declarações-de-parâmetros>)]: <tipo-de-dado>
// Seção de Declarações Internas
inicio
// Seção de Comandos
fimfuncao
```

O *<nome-de-função>* obedece as mesmas regras de nomenclatura das variáveis. Por outro lado, a *<seqüência-de-declarações-de-parâmetros>* é uma seqüência de

```
[var] <seqüência-de-parâmetros>: <tipo-de-dado>
```

separadas por ponto e vírgula. A presença (opcional) da palavra-chave `var` indica passagem de parâmetros por referência; caso contrário, a passagem será por valor.

Por sua vez, *<seqüência-de-parâmetros>* é uma seqüência de nomes de parâmetros (também obedecem a mesma regra de nomenclatura de variáveis) separados por vírgulas.

O valor retornado pela função será do tipo especificado na sua declaração (logo após os dois pontos). Em alguma parte da função (de modo geral, no seu final), este valor deve ser retornado através do comando `retorne`.

De modo análogo ao programa principal, a seção de declaração internas começa com a palavra-chave `var`, e continua com a seguinte sintaxe:

```
<lista-de-variáveis> : <tipo-de-dado>
```

Voltando ao exemplo anterior, no qual calculamos e imprimimos a soma entre os valores 4 e –9, vamos mostrar como isso poderia ser feito através de uma **função sem parâmetros**. Ela também utiliza uma variável local `aux` para armazenar provisoriamente o resultado deste cálculo, antes de atribuí-lo à variável global `res`:

```
funcao soma: inteiro
var aux: inteiro
inicio
// n, m e res são variáveis globais
aux <- n + m
retorne aux
fimfuncao
```

No programa principal deve haver os seguintes comandos:

```
n <- 4
m <- -9
res <- soma
escreva(res)
```

Se realizássemos essa mesma tarefa com uma **função com parâmetros passados por valor**, poderia ser do seguinte modo:

```
funcao soma (x,y: inteiro): inteiro
inicio
retorne x + y
fimfuncao
```

No programa principal deve haver os seguintes comandos:

```
n <- 4
m <- -9
res <- soma(n,m)
escreva(res)
```

**Passagem de Parâmetros por Referência**

Há ainda uma outra forma de passagem de parâmetros para subprogramas: é a passagem por referência. Neste caso, o subprograma não recebe apenas um valor, mas sim o **endereço** de uma variável global. Portanto, qualquer modificação que for realizada no conteúdo deste parâmetro afetará também a variável global que está associada a ele. Durante a execução do subprograma, os parâmetros passados por referência são análogos às variáveis globais. No VisuAlg, de forma análoga a Pascal, essa passagem é feita através da palavra `var` na declaração do parâmetro.

Voltando ao exemplo da soma, o procedimento abaixo realiza a mesma tarefa utilizando passagem de parâmetros por referência:

```
procedimento soma (x,y: inteiro; var result: inteiro)
inicio
result <- x + y
fimprocedimento
```

No programa principal deve haver os seguintes comandos:

```
n <- 4
m <- -9
soma(n,m,res)
escreva(res)
```

**Recursão e Aninhamento**

A atual versão do VisuAlg permite recursão, isto é, a possibilidade de que um subprograma possa chamar a si mesmo. A função do exemplo abaixo calcula recursivamente o fatorial do número inteiro que recebe como parâmetro:

```
funcao fatorial (v: inteiro): inteiro
inicio
se v <= 2 entao
retorne v
senao
retorne v * fatorial(v-1)
fimse
fimfuncao
```

Em Pascal, é permitido o aninhamento de subprogramas, isto é, cada subprograma também pode ter seus próprios subprogramas. No entanto, esta característica dificulta a elaboração dos compiladores e, na prática, não é muito importante. Por este motivo, ela não é permitida na maioria das linguagens de programação (como C, por exemplo), e o VisuAlg não a implementa.

A Linguagem de Programação do VisuAlg (5)

O VisuAlg implementa algumas extensões às linguagens "tradicionais" de programação, com o intuito principal de ajudar o seu uso como ferramenta de ensino. Elas são mostradas a seguir.

**Comando Aleatório**

Muitas vezes a digitação de dados para o teste de um programa torna-se uma tarefa entediante. Com o uso do comando `aleatorio` do VisuAlg, sempre que um comando `leia` for encontrado, a digitação de valores numéricos e/ou caracteres é substituída por uma geração aleatória. Este comando não afeta a leitura de variáveis lógicas: com certeza, uma coisa pouco usual em programação...

Este comando tem as seguintes sintaxes:

| | |
|---|---|
| `aleatorio [on]` | Ativa a geração de valores aleatórios que substituem a digitação de dados. A palavra-chave `on` é opcional. A faixa padrão de valores gerados é de 0 a 100 inclusive. Para a geração de dados do tipo caractere, não há uma faixa pré-estabelecida: os dados gerados serão sempre *strings* de 5 letras maiúsculas. |
| `aleatorio <valor1 > [, <valor2 > ]` | Ativa a geração de dados numéricos aleatórios estabelecendo uma faixa de valores mínimos e máximos. Se apenas `< valor1>` for fornecido, a faixa será de 0 a `<valor1>` inclusive; caso contrário, a faixa será de `<valor1>` a `<valor2>` inclusive. Se `<valor2>` for menor que `<valor1>`, o VisuAlg os trocará para que a faixa fique correta.<br>**Importante:** `<valor1>` e `<valor2>` devem ser **constantes numéricas**, e não expressões. |
| `aleatorio off` | Desativa a geração de valores aleatórios. A palavra-chave `off` é obrigatória. |

**Comando Arquivo**

Muitas vezes é necessário repetir os testes de um programa com uma série igual de dados. Para casos como este, o VisuAlg permite o armazenamento de dados em um arquivo-texto, obtendo deles os dados ao executar os comandos `leia`.
Esta característica funciona da seguinte maneira:

1) Se **não existir** o arquivo com nome especificado, o VisuAlg fará uma leitura de dados através da digitação, armazenando os dados lidos neste arquivo, na ordem em que forem fornecidos.

2) Se o arquivo **existir**, o VisuAlg obterá os dados deste arquivo até chegar ao seu fim. Daí em diante, fará as leituras de dados através da digitação.

3) Somente um comando `arquivo` pode ser empregado em cada pseudocódigo, e ele deverá estar na seção de declarações (dependendo do "sucesso" desta característica, em futuras versões ela poderá ser melhorada...).

4) Caso não seja fornecido um caminho, o VisuAlg irá procurar este arquivo na pasta de trabalho corrente (geralmente, é a pasta onde o programa VISUALG.EXE está). Este comando não prevê uma extensão padrão; portanto, a especificação do nome do arquivo deve ser completa, inclusive com sua extensão (por exemplo, .txt, .dat, etc.).

A sintaxe do comando é:

`arquivo <nome-de-arquivo>`

`<nome-de-arquivo>` é uma constante caractere (entre aspas duplas). Veja o exemplo a seguir:

```
algoritmo "lendo do arquivo"
arquivo "teste.txt"
var x,y: inteiro
inicio
para x de 1 ate 5 faca
   leia (y)
fimpara
fimalgoritmo
```

**Comando Timer**

Embora o VisuAlg seja um interpretador de pseudocódigo, seu desempenho é muito bom: o tempo gasto para interpretar cada linha digitada é apenas uma fração de segundo. Entretanto, por motivos educacionais, pode ser conveniente exibir o fluxo de execução do pseudocódigo comando por comando, em "câmera lenta". O comando `timer` serve para este propósito: insere um atraso (que pode ser especificado) antes da execução de cada linha. Além disso, realça em fundo azul o comando que está sendo executado, da mesma forma que na execução passo a passo.

Sua sintaxe é a seguinte:

| | |
|---|---|
| `timer on` | Ativa o *timer*. |
| `timer <tempo-de-atraso>` | Ativa o *timer* estabelecendo seu tempo de atraso em milissegundos. O valor padrão é 500, que equivale a meio segundo. O argumento `<tempo-de-atraso>` deve ser uma constante inteira com valor entre 0 e 10000. Valores menores que 0 são corrigidos para 0, e maiores que 10000 para 10000. |
| `timer off` | Desativa o *timer*. |

Ao longo do pseudocódigo, pode haver vários comandos `timer`. Todos eles devem estar na seção de comandos. Uma vez ativado, o atraso na execução dos comandos será mantido até se chegar ao final do pseudocódigo ou até ser encontrado um comando `timer off`.

**Comandos de Depuração**

Nenhum ambiente de desenvolvimento está completo se não houver a possibilidade de se inserir pontos de interrupção (*breakpoints*) no pseudocódigo para fins de depuração. VisuAlg implementa dois comandos que auxiliam a depuração ou análise de um pseudocódigo: o comando `pausa` e o comando `debug`.

**Comando Pausa**

Sua sintaxe é simplesmente:

```
pausa
```

Este comando insere uma interrupção incondicional no pseudocódigo. Quando ele é encontrado, o VisuAlg pára a execução do pseudocódigo e espera alguma ação do programador. Neste momento, é possível: analisar os valores das variáveis ou das saídas produzidas até o momento; executar o pseudocódigo passo a passo (com F8); prosseguir sua execução normalmente (com F9); ou simplesmente terminá-lo (com Ctrl-F2). Com exceção da alteração do texto do pseudocódigo, todas as funções do VisuAlg estão disponíveis.

**Comando Debug**

Sua sintaxe é:

```
debug <expressão-lógica>
```

Se a avaliação de `<expressão-lógica>` resultar em valor VERDADEIRO, a execução do pseudocódigo será interrompida como no comando `pausa`. Dessa forma, é possível a inserção de um *breakpoint* condicional no pseudocódigo.

**Comando Eco**

Sua sintaxe é:

```
eco on | off
```

Este comando ativa (`eco on`) ou desativa (`eco off`) a impressão dos dados de entrada na saída-padrão do VisuAlg, ou seja, na área à direita da parte inferior da tela. Esta característica pode ser útil quando houver uma grande quantidade de dados de entrada, e se deseja apenas analisar a saída produzida. Convém utilizá-la também quando os dados de entrada provêm de um arquivo já conhecido.

**Comando Cronômetro**

Sua sintaxe é:

```
cronometro on | off
```

Este comando ativa (`cronometro on`) ou desativa (`cronometro off`) o cronômetro interno do VisuAlg. Quando o comando `cronometro on` é encontrado, o VisuAlg imprime na saída-padrão a informação "Cronômetro iniciado.", e começa a contar o tempo em milissegundos. Quando o comando `cronometro off` é encontrado, o VisuAlg imprime na saída-padrão a informação "Cronômetro terminado. Tempo decorrido: xx segundo(s) e xx ms". Este comando é útil na análise de desempenho de algoritmos (ordenação, busca, etc.).

**Comando Limpatela**

Sua sintaxe é

```
limpatela
```

Este comando simplesmente limpa a tela DOS do Visualg (a simulação da tela do computador). Ele não afeta a "tela" que existe na parte inferior direita da janela principal do Visualg.

Auto-digitação e Sugestão de Digitação

**Auto-digitação**

O VisuAlg tem uma característica para a criação de pseudocódigos que pode aumentar a rapidez da digitação e também diminuir a possibilidade de erros: é a **auto-digitação**. Para utilizar esta característica, basta escrever uma abreviatura da palavra-chave ou do comando a ser diigtado e teclar *Ctrl-Espaço*. O VisuAlg completa então o comando automaticamente, colocando o cursor no ponto adequado para se continuar a digitação (nos exemplos abaixo, este ponto é indicado através de um `*`). Eis a lista de abreviaturas com os respectivos comandos:

! - (Ponto de exclamação) Cria um modelo de pseudocódigo.

```
algoritmo "semnome"
*
inicio
fimalgoritmo
```

# - Cria um cabeçalho de programa.

```
// Algoritmo : *
// Função :
// Autor :
// Data :
```

ale, aof, aon - Inserem os comandos `aleatorio`, `aleatorio off` ou `aleatorio on`, respectivamente.

alg - Insere a linha `algoritmo` e pede a digitação do seu nome.

```
algoritmo "*"
```

arq - Insere o comando `arquivo` e pede a digitação do seu nome.

```
arquivo "*"
```

cof, con - Inserem os comandos `cronometro off` ou `cronometro on`, respectivamente.

dcc - Insere uma declaração de variáveis caracteres.

```
var  * : caractere
```

dcl - Insere uma declaração de variáveis lógicas.

```
var  * : logico
```

dcr - Insere uma declaração de variáveis reais.

```
var * : real
```

deb - Insere o comando `debug`.

eof, eon - Inserem os comandos `eco off` ou `eco on`, respectivamente.

esc - Insere o comando `escreva`.

escl - Insere o comando `escolha` (sem a cláusula `outrocaso`).

```
escolha *
caso
fimescolha
```

esco - Insere o comando escolha (com a cláusula outrocaso).

```
escolha *
caso
outrocaso
fimescolha
```

enq - Insere o comando enquanto.

```
enquanto * faca
fimenquanto
```

fal - Insere a linha fimalgoritmo.

ini - Insere a linha inicio.

int - Insere o comando interrompa.

lep - Insere o comando leia.

```
leia (*)
```

par - Insere o comando para.

```
para * de 1 ate faca
fimpara
```

parp - Insere o comando para com passo.

```
para * de ate passo faca
fimpara
```

rep - Insere o comando repita.

```
repita
*
ate
```

repf - Insere o comando repita com fimrepita.

```
repita
*
fimrepita
```

see - Insere o comando se sem a alternativa senao.

```
se * entao
fimse
```

ses - Insere o comando se completo.

```
se * entao
senao
fimse
```

tim - Insere os comandos `timer on` e `timer off`.

```
timer on
*
timer off
```

tof, ton - Inserem os comandos `timer on` ou `timer off`, respectivamente.

**Sugestão de Digitação**

A sugestão de digitação é disponibilizada através das teclas *Ctrl-J*. Basta começar a digitação de uma palavra e teclar *Ctrl-J* para que o VisuAlg mostre uma lista com sugestões de palavras-chave que completam o que foi digitado. Para escolher, é necessário dar um duplo-clique sobre a opção desejada, ou então selecioná-la com as setas e teclar *Enter*. Se o usuário continua escrevendo depois que o VisuAlg apresentou a lista de sugestões, o programa continuará procurando palavras que ainda complementem o que foi digitado. Ao se teclar *Esc* ou clicar "fora da lista", ela desaparece.