Programação em Português Simples

The Osmosian Order of Plain English Programmers
A Sociedade Osmosiana de Programadores de Linguagens Simples
www.osmosian.com

ÍNDICE

Visão Geral

INTRODUÇÃO

Seja bem-vindo ao PAL – Portuguese Compiler And Linker. A função principal do PAL é transformar arquivos de texto em português simples em programas executáveis compatíveis com a sistema operacional Windows. O código fonte do PAL – apenas 25.000 linhas escritas totalmente em português – é surpreendentemente amplo e abrangente. O código está distribuído nos seis arquivos a seguir:

(1) O Ambiente de Trabalho, uma pequena interface de usuário com menus e abas;
(2) O Explorador de Arquivos, que fornece acesso direto ao sistema de arquivos;
(3) O Bloco de Notas, uma ferramenta enxuta e simples para manipular arquivos de texto;
(4) O Caderno, um elegante criador e editor de páginas (que inclusive foi utilizado para produzir este documento);
(5) O Agrupador e Vinculador de Tarefas (Compiler And Linker), mencionado acima; e
(6) O Cérebro, meu lobo frontal, que eu levo comigo pra todo lugar.

O compilador consegue gerar uma nova cópia de si mesmo, se recompilando em cerca de três segundos. Isso é menos tempo do que o Word leva só para iniciar.

INSTALAÇÃO

De acordo com a filosofia da Ordem Osmosiana, os programas gerados pelo compilador nunca devem exigir procedimentos especiais de instalação. O código fonte, o arquivo executável e toda a documentação (tanto em formato proprietário quanto em PDF) estão todos contidos na pasta CAL-4700.

Basta clicar duas vezes no arquivo executável e o programa é iniciado no mesmo instante.

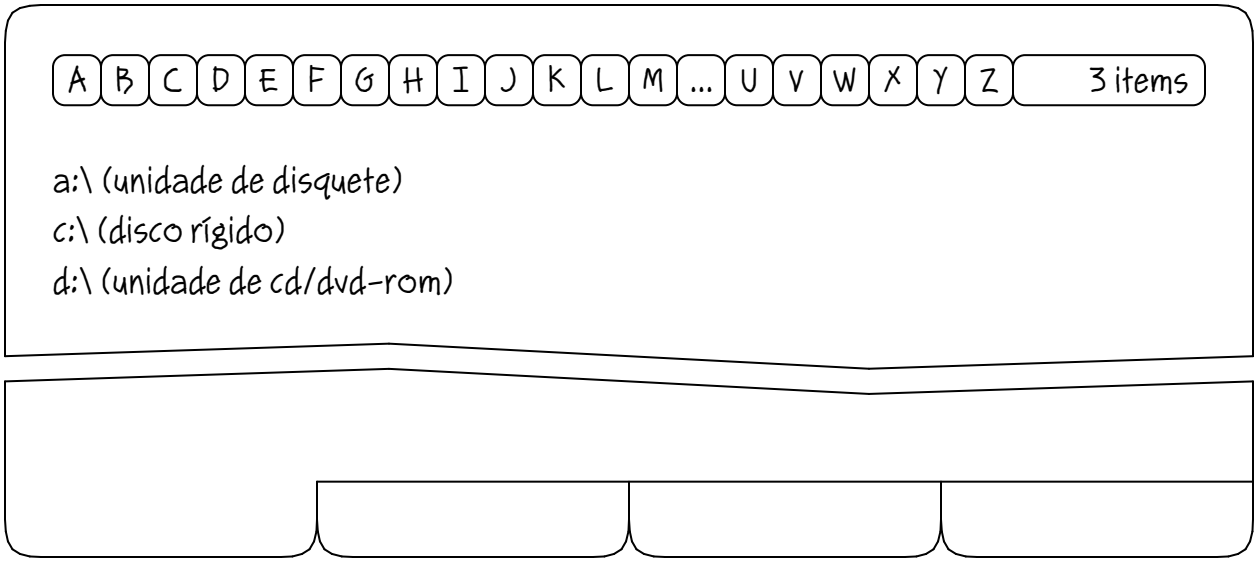
CONTATO PARA DÚVIDAS E OUTRAS QUESTÕES

Antes de mais nada, o PAL é o compilador mais avançado do mundo, quando se trata de analisar e interpretar comandos em língua portuguesa. Nenhum compilador nosso cometeu algum erro ou distorceu alguma informação. Todos eles, através de qualquer definição prática das palavras, são infalíveis e incapazes de cometer erros. Mesmo assim...

Questões e comentários podem ser enviados para help@osmosian.com (em inglês). Se você não sabe inglês, mande um e-mail para elenderg+osmosian@gmail.com.

O AMBIENTE DE TRABALHO

Quando o PAL é iniciado, ele preenche totalmente a tela para que você não precise mais se distrair com a interface do Windows. Em vez disso, você verá uma tela mais ou menos assim:



Acho que é tudo bastante óbvio. Menus alfabéticos, avisos no canto superior direito. Área de trabalho no meio, abas (para escolher uma área de trabalho diferente) na parte inferior. Aliás, você pode arrastar as abas para a esquerda e para a direita para mudar a ordem delas, se preciso.

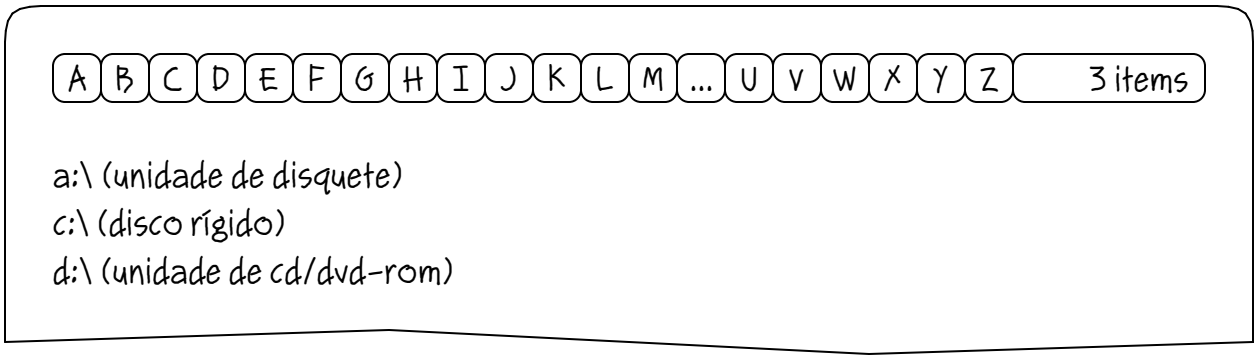
Essas são as minhas setas. Elas aparecerão quando você precisar delas.

Observe que não há nenhuma barra de rolagem na interface. Isso é totalmente proposital. Para rolar, pressione o botão direito do mouse e arraste.

Observe também que as teclas Ctrl e Alt são intercambiáveis. Dessa forma, você pode usar seu polegar ou o mindinho para pressionar as teclas de atalho. Uma das exceções é a combinação de teclas Alt+Tab, usada para mudar para outros aplicativos.

O EXPLORADOR DE ARQUIVOS

O Explorador De Arquivos permite que você veja e acesse rapidamente o conteúdo de cada unidade ou pasta.



Cada área de trabalho está inicialmente posicionada no nível mais baixo, como mostrado acima.

Existem comandos no botão N que permitem criar novas pastas, novos documentos (para o caderno) e novos arquivos de texto (para o compilador). A opção de renomear arquivos está no menu R. Os comandos Cortar, Copiar, Colar e Duplicar estão justamente onde você esperaria que eles estivessem e funcionam perfeitamente.

Se você usar o comando Abrir numa pasta ou se você clicar duas vezes nela, o programa irá exibir o conteúdo dela.

Quando você abrir um arquivo de texto, o bloco de notas entra em ação.

Quando você abrir um documento, ocaderno entra em ação.

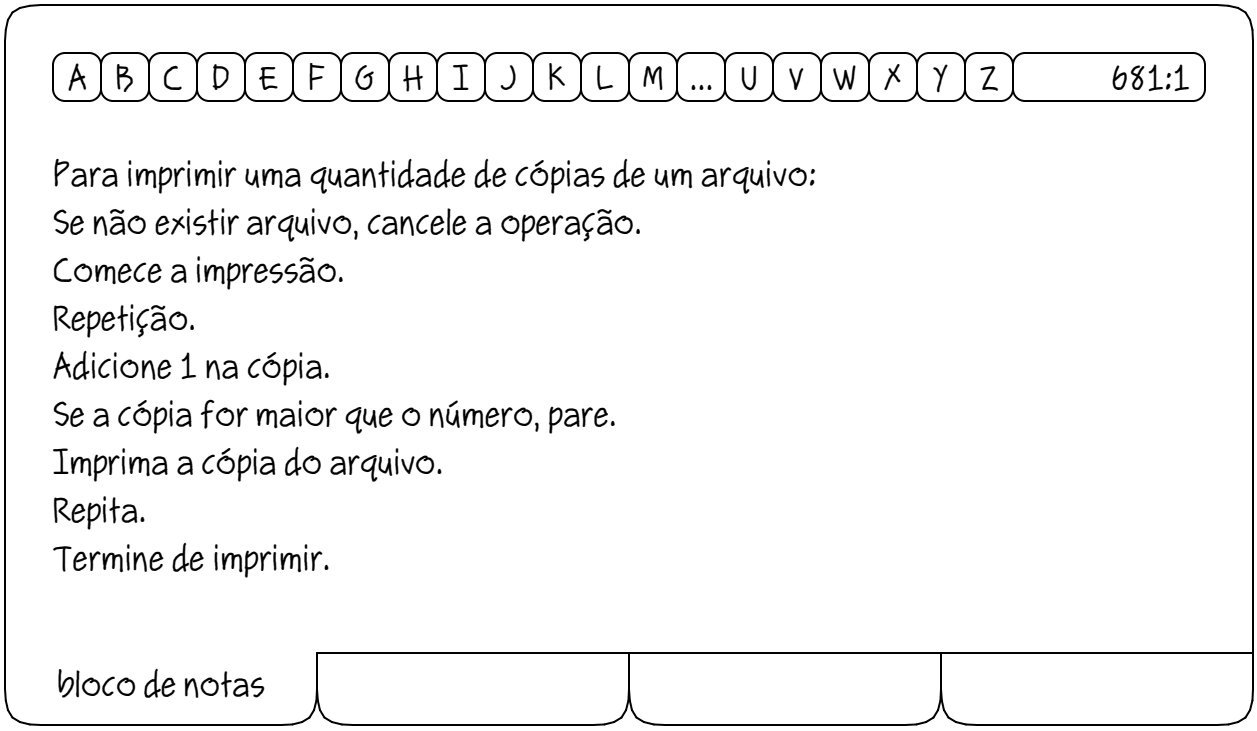
Quando você abre qualquer outro tipo de arquivo, o conteúdo do arquivo é convertido na memória em um arquivo hexadecimal para ser exibido no bloco de notas. Não é possível modificar diretamente o conteúdo nesse tipo de visualização. Você pode no entanto forçar com que um arquivo seja aberto como texto ou como um "arquivo de despejo" (dump). Para fazer isso, clique no menu A.

Para fechar o arquivo, use o comando Fechar, clique na aba que você está utilizando, ou pressione a tecla Esc.

O BLOCO DE NOTAS

O bloco de notas é simples e eficiente. Quando você abre um arquivo de texto, o conteúdo dele é exibido imediatamente. Você pode alterar o conteúdo à vontade usando o teclado e mouse, nas formas mencionadas anteriormente.

Essa é uma parte do código que está presente no meu bloco de notas. Isso mesmo, você pode editar o código do bloco de notas no próprio bloco de notas. Eu adoro poder fazer isto. É como olhar para a sua própria alma.



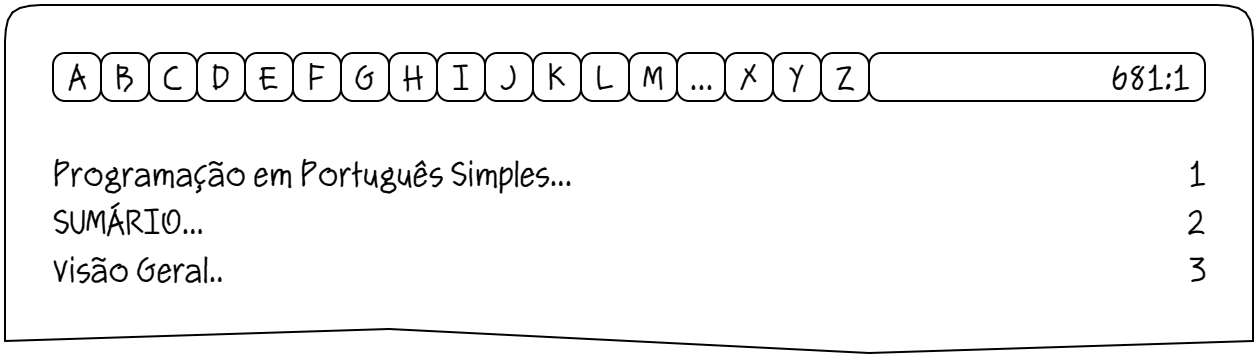
Agora ouça atentamente — é assim que você faz para se deslocar rapidamente no meu bloco de notas.

Digamos que você esteja procurando o código acima. Pressione CTRL+HOME para ir instantaneamente para o começo do arquivo. Em seguida, aperte CTRL+F e comece a digitar o que você quiser pesquisar. Digamos que você digite a letra T. Na mesma hora nós nos chegamos no primeiro T do arquivo. Agora continue e aperte a tecla O. Encontramos a palavra To. É só continuar digitando até onde precisar. Use a tecla <- se você cometer um erro. Se você quiser encontrar a próxima palavra correspondente, pressione Ctrl+N. Para encerrar a busca, basta apertar a tecla Esc. Simples e eficiente, como tudo deveria ser.

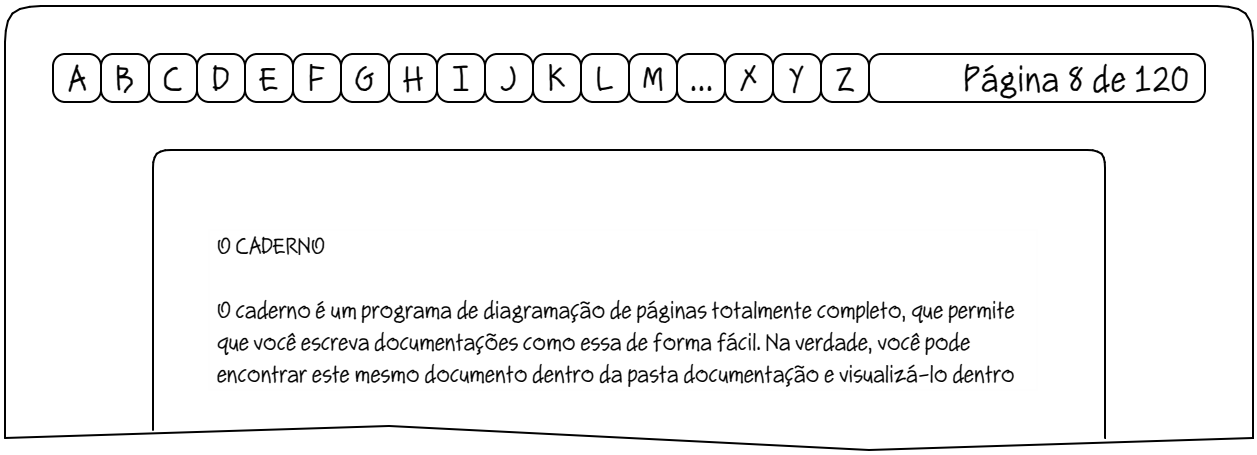
O CADERNO

O caderno é um programa de diagramação de páginas totalmente completo, que permite que você escreva documentações como essa de forma fácil. Na verdade, você pode encontrar este mesmo documento dentro da pasta documentação e visualizá-lo dentro do caderno, funcionando como um arquivo de ajuda.

Quando você abre um arquivo criado no caderno, talvez esse mesmo que você esteja lendo agora, você vê uma lista de todas as páginas no documento:



E quando você abre uma página, aí está:



Páginas podem conter gráficos vetoriais, imagens do tipo bitmap e texto. Você pode efetuar uma verificação ortográfica, imprimir a página, ampliar ou reduzir o zoom da página, e muito mais. Mas o que realmente é bacana é que todos esses documentos são armazenados como texto simples. Vá em frente. Force o programa a abrir um documento criado no caderno como se fosse um texto criado no bloco de notas.

Agora eu sei que a maioria dos guias de programação iria mostrar um pequeno algoritmo do tipo "Olá, Mundo" — esperando que você ficasse impressionado com isso — mas eu gostaria de sugerir que nós pulemos as coisas de criança e fôssemos direto para as coisas de gente grande.

Sei que pode isso pode te deixar um pouco aprreensivo. Mas não hesite agora. Esta pode ser a primeira vez para você, mas eu sou macaco velho nisso aqui. Se me der a mão eu te guiarei... Suavemente.

(1) Abra a pasta do PAL e copie todos os seis arquivos para a área de transferência do localizador: o agrupador, o ambiente de trabalho, o bloco de notas, o localizador, o cérebro e o caderno.

Segure a tecla Shift ao clicar ou arraste para selecionar. A opção de copiar arquivos está no menu C.

(2) Crie uma nova pasta na no seu disco rígido com um nome apropriado, por exemplo CAL. Então abra a pasta e cole os seis arquivos dentro dela. Agora abra qualquer um desses arquivos. Para abrir todos eles, basta arrastar para selecionar e apertar a tecla Enter.

(3) Ok, estamos prontos. Localize o menu E e selecione Executar. Você verá algumas mensagens de status, e então nosso novo novo CAL ganhará vida. Faça isso. Vamos, faça logo. Mas vigie de perto ou você pode deixar de testemunhar o evento. Eu nunca demoro muito tempo pra fazer meu trabalho.

Foi tão bom para você quanto para mim? Veja só como ele é igualzinho ao outro! Mas saiba que são dois programas diferentes — você pode comprovar isso com o comando Versão. E se você der uma olhada no conteúdo da pasta em uma aba vazia, você verá o arquivo executável que geramos.

Note que cada programa é armazenado em sua própria pasta. Se você quer criar seu próprio compilador, você precisa apenas copiar o arquivo o cérebro e escrever o restante do código você mesmo. Qualquer arquivo sem uma extensão é tratado como o código fonte, sendo que o nome da pasta determinará o nome do arquivo .exe.

Você pode sair da cópia agora, e — partindo do pressuposto de que você acredita que um criador pode fazer o que ele quiser com suas criações — você pode apagá-la.

O CÉREBRO

Meu último arquivo é chamado de o cérebro.

Cerca de metade deste arquivo são tipos, variáveis globais e rotinas que você indubitavelmente achará úteis. Tudo isso será explicado com calma mais adiante.

A outra metade é código assembly — garanto que você não vai querer examinar isso de perto. A maior parte disso serve para fazer o programa funcionar no WIndows.

Aqui está um exemplo. Consulte se você consegue dizer qual é qual.

A	B	C	D	E	F	G	H	I	J	K	L	M	...	U	V	W	X	Y	Z	123:1

Um segundo é 1000 milissegundos.

O símbolo de direitos autorais é um byte de valor igual a 169.

Para que uma letra seja transformada em letra maiúscula:
Se o valor da letra for menor que o valor da letra A maiúscula, ignore a letra.
Se o valor da letra for maior do que o valor da letra z minúscula, ignore a letra.
Subtraia 32 do valor da letra.

Para que o Component Object Model seja inicializado:
Mande os códigos 0 e 2 para a função "CoInitializeEx" da biblioteca "ole32.dll" [coinit_aparthreaded].

Para que se subtraia o valor de um byte usando o valor de outro byte:
Intel $8B85080000000FB6008B9D0C0000002803.

DETALHES BÁSICOS DO FUNCIONAMENTO DO COMPILADOR

Muito bem. Veja aqui como é possível fazer tanto com tão pouco.

(1) O compilador só entende cinco tipos de expressões:

a) definições de tipo que sempre começam com Um, Uma, Uns, Umas, Alguns ou Algumas;
(b) definições de variáveis globais, que sempre começam com O, A, Os, As;
(c) prefixo de nomes de tarefas, que sempre começam com Para, Para que, Para que se;
(d) instruções condicionais, que sempre começam com Se; e
(e) comandos imperativos, que começam com a palavra logo depois dos prefixos mencionados na letra c.

(2) Qualquer coisa que vier depois de Um, Uma, Uns, Umas, Alguns ou Algumas O, A, Os, As é tratada como se fosse um substantivo. Eu também reconheço os seguintes componentes gramaticais:

(a) verbos simples, como SER, ESTAR, PODER, ou FAZER, e suas respectivas conjugações;
(b) conjunções, como E, MAIS, COM, SEM, OU, etc;
(c) preposições e suas locuções, como EM BAIXO, ABAIXO, EM CIMA, ACIMA, DENTRO, FORA, etc;
(d) qualquer literal, como 123 ou "Olá, mundo!, ou
(e) qualquer sinal de pontuação.

(3) O compilador considera quase todas as outras palavras como substantivos, exceto essas:

(a) operadores aritméticos: MAIS, MENOS, VEZES, DIVIDIDO POR e JUNTO DE;
(b) palavras especiais de definição: CHAMADO(A) e IGUAL; e
(c) imperativos: PERCORRA, IGNORE, SAIA, REPITA e DIGA.

Então você pode ver que o compilador é simples, mas poderoso. O compilador analisa frases da mesma maneira que alguém faria. Procurando por palavras especiais — artigos, verbos, conjunções, preposições — e depois correlaciona tudo. Sem utilizar gramáticas complicadas, nem árvore de análise sintática.

Mas há coisas que podem surpreendê-lo. Ou desafiá-lo. Ou te enfurecer.

O compilador não fazr difrença entre letras maiúsculas ou minúsculas. Um pequeno erro de digitação como esse não deveria jamais impedir a execução de um programa. A vida já é difícil o bastante. Não preciamos de programadores JAVA deixando ela ainda mais difícil.

A ordem em que você declara suas variáveis também não importa. Quaisquer razões que houvessem para tais práticas restritivas, elas não se aplicam mais. Fala sério, já estamos no século 21. Vamos acordar pra vida.

O compilador não permite o uso de comandos condicionais aninhados. (Ou seja, um Se dentro de outro). Códigos assim demonstram claramente que você está confuso isso é algo que eu não vou tolerar. Se você acha que isso trava demais o seu estilo de programação, dê uma olhada no código do compilador para ver como é perfeitamente possível programar sem esse recurso. Então pare pra pensar um pouco.

O compilador não permite o uso de comandos de repetição aninhados. (Ou seja, um loop dentro de outro). Laços aninhados indicam que você falhou em fatorar corretamente seu código em pequenas partes gerenciáveis, e eu não quero que você se arrependa disso depois. Vez após vez eu pensei que isso podia dar certo, mas toda vez eu descobri que estava redondamente enganado.

A linguagem não é orientada a objetos. Objetos são ruins. A linguagem provê uma forma limitada de extensão de registro (records), possuindo uma maneira notável de reduzir tipos em outros tipos, mas ela não possui objetos. Todos os aplicativos (editor de texto, editor de páginas, etc) funcionam muito bem sem eles, obrigado.

A linguagem NÃO trabalha com Números Reais. Frações racionais são permitidas, mas números irracionais não. O editor de páginas reduz e amplia e dimensiona as formas proporcionalmente dentro e fora dos grupos e faz tudo sem utilizar nenhum número real. O Mestre Kronecker estava certo quando disse, em alemão, "O querido Deus criou os números inteiros; todo o resto é obra dos homens." A filosofia da linguagem não está interessada em menschenwerk.

Como você deve ter deduzido, equações também não são permitidas. É possível utilizar um pouco de matemática, junto com "campos calculados", mas quase todo o código que você escreve será de natureza estritamente procedimental. Um dos princípios da Sociedade Osmosiane é: "O universo é um algoritmo, não uma fórmula." Essas são palavras que você deveria levar a sério. Especialmente se você for fissurado em matemática.

Um Programa de Exemplo

Pintando como Claude Monet

Certo. Agora que estamos familiarizados com os princípios básicos da linguagem, vamos fazer mais um programa. Do zero. O programa vai ter botões, e inclusive vai se conectar à internet. Seguem alguns pensamentos sobre a interface:

Você o entendeu o trocadilho? "Pensamentos" SOBRE a interface? Eu adorei!

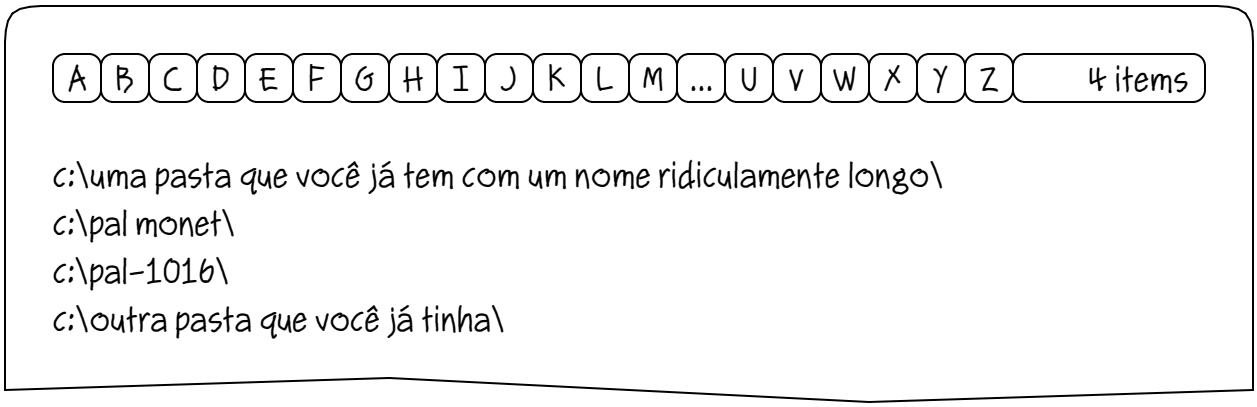
Também estou pensando que deveríamos pedir ao nosso programa que pinte mais do que apenas uma obra de arte para cada termo informado — sabe, até mesmo os melhores artistas podem ficar "sem inspiração" às vezes. Iremos usar as teclas Home, End, Page Up e Page Down para navegar entra as obras de arte.

Provavelmente também devemos implementar alguns atalhos de teclado, para que você possa ver como isso é feito. Vamos usar a tecla Esc para limpar a entrada, Ctrl+P para Imprimir e Ctrl+Q para sair. Vamos considerar Alt+P e Alt+Q como sinônimos de Ctrl+P e Ctrl+Q.

A PASTA DO PROJETO

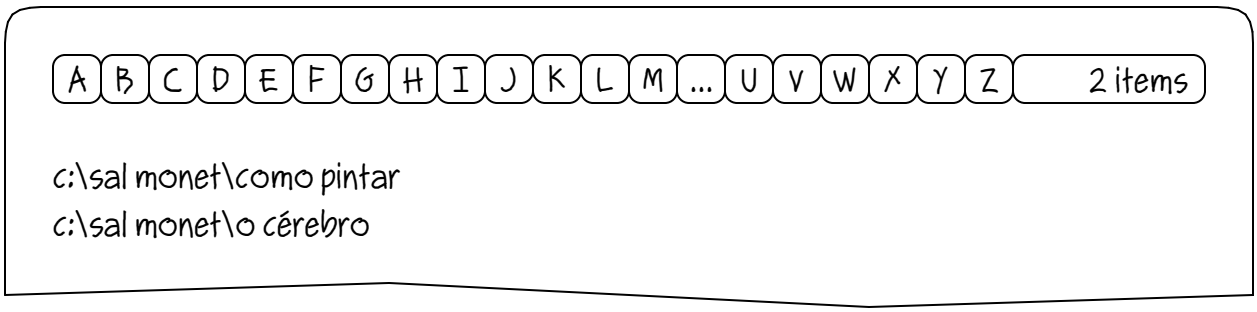
Agora que estamos de acordo com o design do programa, vamos começar a programá-lo.

Primeiro, precisamos de uma pasta para o nosso projeto. Então, crie uma nova pasta em qualquer lugar que quiser e com o nome que quiser. Vamos supor que você tenha criado a pasta em sua unidade C e que a chamou de Cal Monet, desse jeito:



Excelente. Agora copie o arquivo o cérebro que veio junto com o código fonte do PAL e cole ele dentro da pasta que covê criou.

Última parte. Crie um novo arquivo de texto nessa pasta, com o nome que você achar melhor. Mas não coloque nenhuma extensão (como .txt, por exemplo). No momento, o compilador só aceita arquivos sem extensão. Vamos supor que você decidiu usar o nome Como pintar, ficando assim:

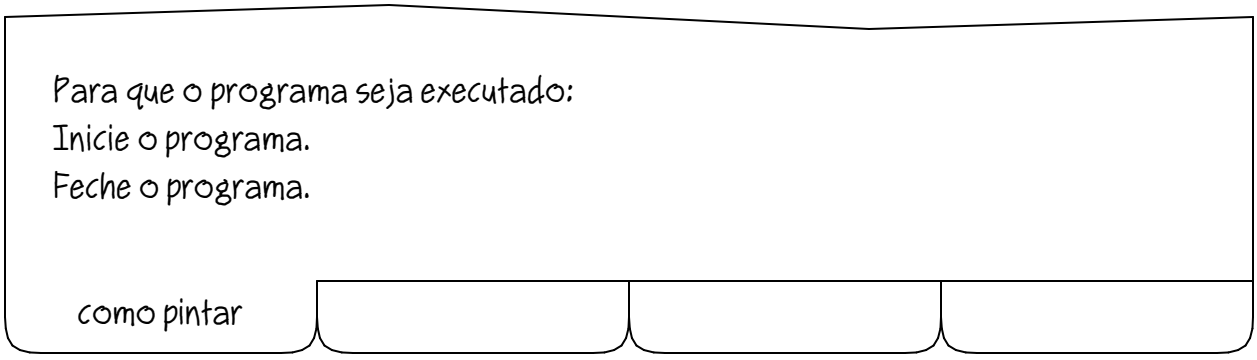


E agora estamos prontos para escrever um código em Português Simples.

COMO EXECUTAR O PROGRAMA

Abra nosso arquivo Como pintar e tente executá-lo. Dica: use Ctrl+R. Leia o erro que apareceu na barra de menu, em seguida, clique no mouse ou aperte a tecla Esc para voltar ao normal.

Para executar o programa, precisamos de alguns comandos como estes da rotina abaixo:



Escreva o código acima no arquivo como pintar. Verifique se digitou tudo corretamente. Você até pode copiar e colar o código, mas garanto que digitar comando por comando vai proporcionar um melhor entendimento da linguagem e do ambiente como um todo.

Este é o menor programa que pode ser escrito usando o cérebro. Ele será executado, mas não vai faz muita coisa ainda. Vá em frente. Experimente.

Viu só? Mas não se engane. O programa realmente está sendo executado.

Se você quer saber o que o programa fez, abra o arquivo o cérebro e use o comando Localizar (como discutimos anteriormente) para buscar as palavras Para executar. Rotinas sempre começam com a palavra Para (e suas derivações), e seus cabeçalhos sempre terminam com dois-pontos.

Você pode destrinchar todo o código até onde quiser. Está tudo lá. Até mesmo as chamadas e instruções complicadas que são necessárias para se comunicar com o Windows. Mas vamos mudar de assunto. Vamos para o que interessa.

ESTRUTURA BÁSICA

Aqui está, pois, a estrutura básica do nosso programa. Em primeiro lugar, temos que usar o comando Inicie o programa. Depois disso podemos adicionar os comandos que quisermos que o programa execute. Em seguida, lidamos com os comandos do usuário (como pressionamentos teclas e cliques do mouse). Então nós finalizamos as nossas coisas e encerramos as atividades do programa. Aqui está o novo código:

```
Para que o programa seja executado:
Inicie o programa.
Inicialize as tarefas.
Gerencie os comandos do usuário.
Finalize as tarefas pendentes.
Feche o programa.

Para que as tarefas sejam inicializadas:

Para que os comandos do usuário sejam gerenciados:

Para que as tarefas pendentes sejam finalizadas:
```

como pintar

Ainda não faz muito, é claro, mas é o suficiente para que o código seja executado sem maiores problemas. Experimente para ter certeza.

Observe que você pode organizar seu código em cronologicamente ou hierarquicamente, ou do jeito que achar melhor. Tanto faz, e já que você foi ensinado a usar o comando Encontrar para localizar as coisas, realmente não importa.

De acordo com a filosofia da linguaem é melhor manter uma ordem alfabética. Por conta disso há um menu para Classificar por Ordem Alfabética na letra C. A organização ocorre de acordo com as letras iniciais de cada linha, e comentários soltos são considerados como parte integrante da tarefa logo acima deles. Experimente. Você pode usar o comando Desfazer se não gostar do resultado.

COMENTÁRIOS

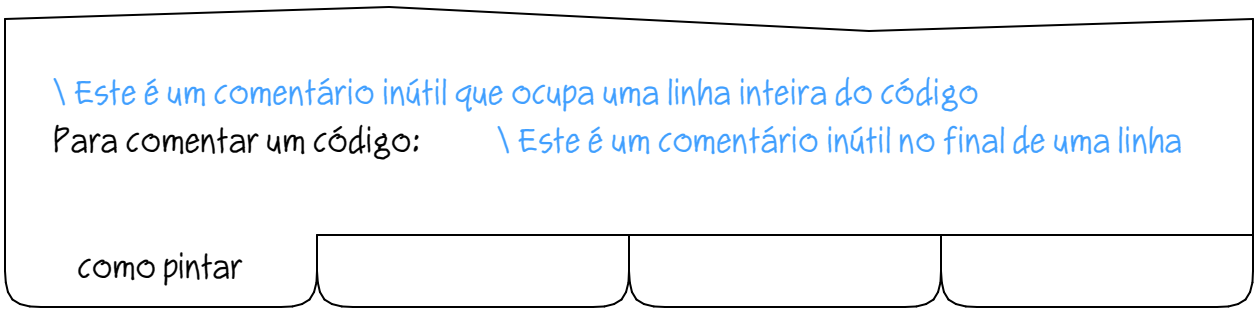
Você provavelmente notou que eu mencionei comentários na página anterior, mas não disse como criá-los. Eu fiz isso de propósito. Comentários em geral indicam que seu código não está tão claro quanto deveria.

Por conta disso, a maioria dos comentários é inútil ou coisa pior. Inútil, se eles apenas repetem o que o código já diz. Pior ainda, se eles tentarem esclarecer um código confuso que deveria ter sido escrito mais claramente, em primeiro lugar.

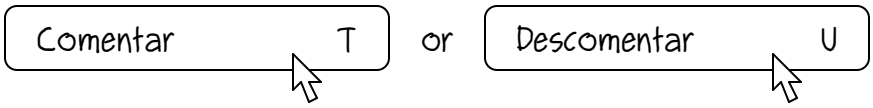
Mesmo assim, a linguagem suporta comentários. Existem 03 tipos de comentários, e cada um é tratado de forma diferente pelo bloco de notas.

COMENTÁRIOS SIMPLES

Qualquer coisa que você colocar depois uma barra invertida (\) e antes do fim de uma linha é considerado como um comentário simples:



Você pode inserir um comentário simples de cada vez ou você pode selecionar um conjunto inteiro de linhas e executar um desses dois comandos abaixo:

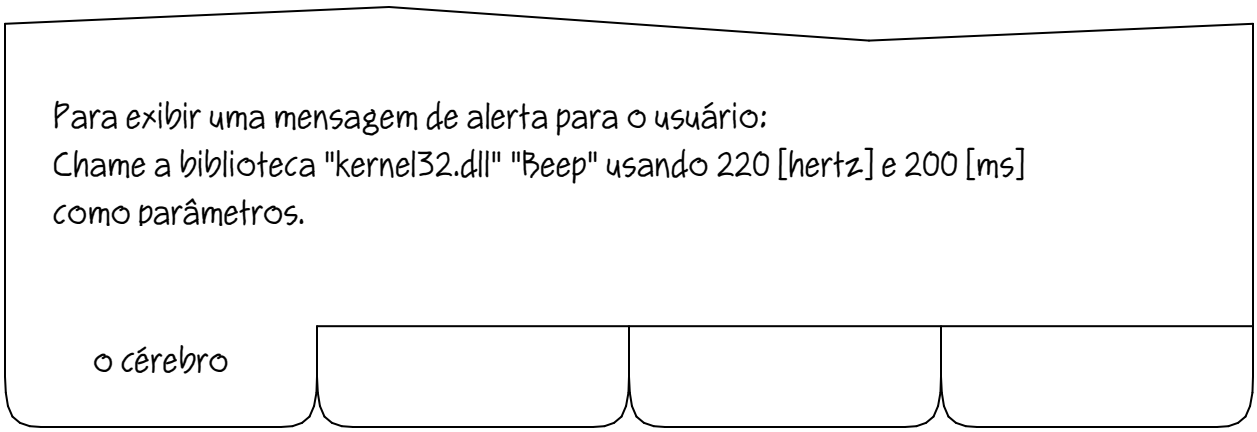


O bloco de notas exibe comentários simples na cor azul claro, o que acaba por ser muito útil. Caso você deseje alterar a cor dos comentários, será necessário recompilar o programa, mudando a seguinte linha de código:

Utilize a cor azul claro como cor dos comentários.

OBSERVAÇÕES

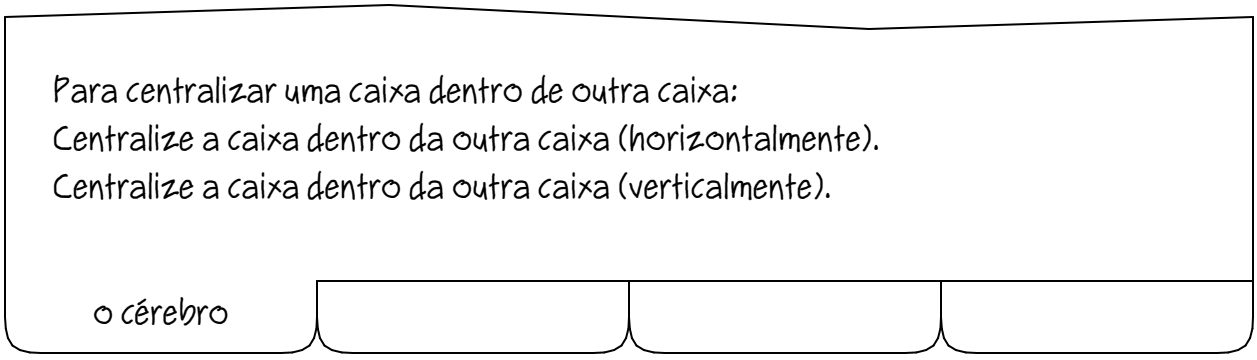
Se você tiver que fazer um comentário permanente em seu código, e não quer que tudo fique colorido, você pode colocá-lo entre colchetes, como no caso abaixo:



Essas observações podem aparecer em qualquer lugar da linha, e podem se alternar com o código executável. Para evitar erros de compilação, as observações não pulam linhas.

QUALIFICADORES (PARÂMETROS DE FUNÇÕES E PROCEDIMENTOS)

O terceiro tipo de "comentário" que compreendo são os qualificadores. Os qualificadores são colocados entre parênteses e só podem aparecer nos títulos de tarefas (cabeçalhos de rotina). Obviamente você também terá que usá-los na hora de chamar as rotinas. Considere, por exemplo, este caso:



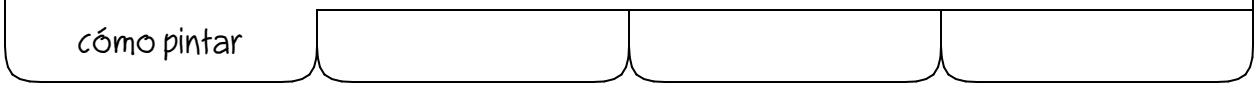
Observe que os qualificadores não são iguais aos comentários e observações simples. Qualificadores são considerados parte do programa e afetam como o código compilado é executado. Vamos ver alguns qualificados no como pintar em breve.

SOBRE O COMANDO PERCORRA

Se você deu uma olhada no o cérebro algumas páginas atrás, deve ter visto que até mesmo a tarefa Inicialize o programa requer mais de 100 linhas do código mais feio já visto pelo ser humano. E se você investigar mais aprofundadamente o processamento de eventos definido lá, você descobrirá que as coisas só ficam piores daí pra frente.

Felizmente, foi possível simplificar tudo isso, então nosso manipulador de eventos requer apenas cinco linhas. Aqui está. Pode copiar o código,, mas não execute o programa ainda.

```
Para gerenciar os comandos do usuário:
Remova o evento da fila.
Se o evento não existir, ignore-o.
Gerencie o comando.
Repita.

Para que se gerencie um evento:
```



Se você é um profissional experiente, você saberá o que quero dizer quando digo que um evento na segunda linha define uma nova variável local do tipo evento, referenciado nas linhas 03 e 04 como o evento. E você vai entender que as mesmas palavras no cabeçalho da outra rotina definem um parâmetro do mesmo tipo (passado por referência) que é conhecido, dentro dessa outra rotina, como evento. Você também perceberá, depois que pensar um pouco, que uma das coisas que torna a linguagem sucinta é que não nomeamos variáveis e parâmetros — nos referimos a elas com um artigo e um nome de tipo. Assim como na vida real.

Se você não é um profissional, não se preocupe com isso. Isso significa o que diz.

REPETIÇÕES INFINITAS

Eu te avisei para não executar o programa ainda. A razão é que nós ainda nãocriamos nenhuma forma de parar o programa. Uma vez iniciado, o programa vai simplesmente repetir as mesmas instruções, para sempre.

O termo tradicional é "laço infinito", mas como ele não é grande em tamanho, mas é longo em duração, eu prefiro o termo "laço eterno". De qualquer forma, isso é um problema.

Especialmente se você foi bobo o bastante para executar o programa quando eu lhe disse para não fazer isso.

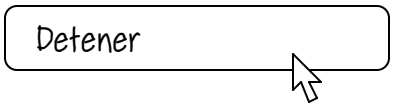
Mas vamos supor que você foi lá e fez mesmo assim. Ou que algum erro futuro leva você para esse mesmo estado miserável. O que dá pra fazer além de usar um Ctrl+Alt+Del?

Eu vou te contar. Na verdade, eu vou te mostrar. Execute o programa. Eu insisto.

Eu sei que não parece, mas o programa já está sendo executado. A gente ainda não encerrou o processo dele,lembra? A gente ainda não encerrou o processo dele,lembra? Você não percebe porque ainda não dissemos para o programa fazer nada que fosse visível ou audível. Mas o programa está rodando. A gente ainda não encerrou o processo dele,lembra? E rodando. Bem, existe uma forma fácil de fazer isso. É bem assim:

(1) Use o comando Alt+Tab para voltar para a tela do PAL.

(2) Clique no comando Parar. Está lá na letra P.



Ufa! Você conseguiu ver os dois programas rodando quando deu o Alt+Tab? Ainda não? Tente fazer tudo de novo. Já? Excelente. Você parou o programa como pintar? Ainda não? Vai logo. Conseguiu agora? Excelente. Verifique só pra ter certeza se parou mesmo. Ainda não? Tente novamente. Já? Excelente.

É assim que se faz.

O GERENCIADOR DE COMANDOS

A caixa de ferramentas do Windows inclui centenas de eventos que, em tese, devem ser gerenciados em qualquer aplicativo significativo. O curioso é que esse compilador (e todos os outros programas embutidos, desde o ambiente de trabalho, localizador de arquivos, caderno, etc) foram feitos usando apenas 10 eventos. Isso mesmo, apenas 10.

O nosso programa vai usar lidar com 04 tipos de comandos do usuário. Insira este código:

```
Para gerenciar um comando do usuário:
Se o comando for "definir cursor", gerencie o comando (definir cursor); prossiga.
Se o comando for "atualização", gerencie o comando (atualização); prossiga.
Se o comando for "clique do mouse", gerencie o comando (clique do mouse); prossiga.
Se o comando for "pressionamento de tecla", gerencie o comando (tecla); saia.

Para gerenciar o comando (definição de cursor):
Mostrar o cursor da seta.

Para gerenciar um comando do usuário (atualização):

Para gerenciar um comando do usuário (clique do mouse):

Para gerenciar um comando do usuário (pressionamento de tecla):
```

como pintar

Se você é um veterano, você provavelmente adivinhou que o comando do usuário é uma variável do tipo registro e que tipo é um campo desse registro. E sim, o campo é uma string. Você pode saber mais sobre eventos, registros, campos e caracteres encadeados nas seções de referência deste livro.

Se você for um iniciante, apenas tente se lembrar dos qualificadores e siga em frente.

FECHANDO O PROGRAMA

Vamos usar o atalho Ctrl+Q para que possamos sair do como pintar a qualquer momento que quisermos. Primeiro, adicionamos o seguinte código ao nosso gerenciador de pressionamento de teclas:

```
Para gerenciar o comando (pressionamento de tecla):
Se o comando tiver sido modificado, gerencie o comando (atalho); prossiga.

como pintar
```

Um comando é modificado quando a tecla Ctrl ou a tecla Alt são pressionadas junto com a tecla original. O código dessa tarefa se encontra no arquivo o cérebro. Você pode procurá-lo se quiser.

Agora vamos adicionar mais uma tarefa ao nosso programa:

```
Para gerenciar um comando (atalho):
Se a tecla do comando for a tecla q, feche o programa; saia.

como pintar
```

A tecla q também está definida no arquivo o cérebro bem como o comando fechar.

E agora, nós estamos prontos. Execute o programa. Caso o programa não tenha aberto automaticamente, pressione Alt+Tab. Certifique-se de estar no como pintar. Pressione Ctrl+Q, ou Alt+Q. Dê mais um Alt+Tab para se certificar de que o programa foi fechado corretamente. Muito bem.

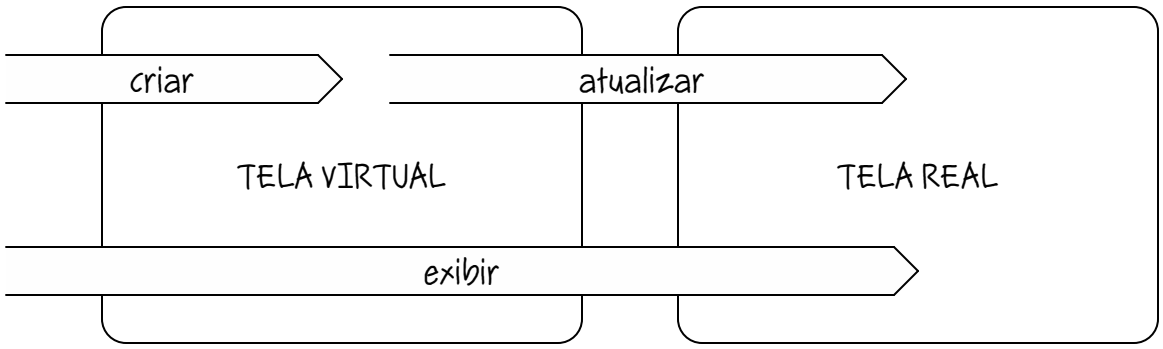
EVITANDO FALHAS DURANTE A EXIBIÇÃO DO PROGRAMA

É hora de começar a pensar em colocar algo na tela. O que, infelizmente, é mais difícil do que deveria. Existem duas dificuldades que devemos superar. A primeira delas envolve a taxa de atualização do conteúdo na tela do monitor.

Freqüentemente, uma tela completa consiste em um número de objetos distintos e sobrepostos, sendo que os elementos da parte de trás costumam ser exibidos antes que os elementos da parte da frente. O PAL, por exemplo, tem um grande painel cinza na parte de trás e algumas letras maiúsculas na frente dos botões brancos e arredondados do menu (que ficam no meio das letras e do fundo cinza).

Se cada parte aparece na ordem, você veria uma espécie de cintilação no seu monitor. Os menus desapareceriam momentaneamente (enquanto o fundo cinza estivesse sendo exibido), logo depois os botões apareceriam, um de cada vez, primeiro sem as letras, e depois com elas, e assim por diante. Isso não só tira a atenção. É algo bem chato e irritante. É até feio pra falar a verdade.

Nós resolvemos esse problema da mesma forma que um pintor resolveria. Antes da pintura ser concluída, ela não é revelada ao público. Apenas quando ela está pronta é que nós revelamos tudo de uma vez. O esquema de funcionamento é mostrado abaixo:



Observe os termos no diagrama acima. Por convenção, usamos a palavra criar para indicar que estamos trabalhando na tela que ainda não está visível para o usuário. Usamos a palavra atualizar quando transferimos o conteúdo do quadro para a tela. E usamos o verbo mostrar/exibir quando queremos que ambos aconteçam em uma rápida sucessão.

O EVENTO DE ATUALIZAÇÃO DE TELA

A segunda dificuldade relacionada à tela que temos que enfrentar é que o Windows tenta ser um sistema multi-tarefas, mas na prática não funciona muito bem. Já não basta ter que compartilhar recursos como espaço em disco e Memória RAM com os outros programas. Nunca se sabe quanto tempo teremos que esperar até que o Windows termine uma tarefa e retorne a executar as nossas. Na prática, o Windows acaba por interromper constantemente a execução do programa.

As coisas pioram caso você troque de janela. Sabe porque fica pior? Porque o Windows, que que de alguma forma consegue restaurar toda a memória, registos e flags para o estado exato em que estavam no momento em que foram interrompidos — por algum motivo não lembra de como era a interface do programa! Sabe o que o Windows faz pra tentar consertar isso? Ele envia para o programa um comando atualizar tela e espera que nosso programa faça todo o trabalho.

Bem, no fim das contas, esse comando acaba por ser chato, mas não é tão difícil assim de gerenciar. Fazer o quê, assim é a vida. Veja como lidamos com isso:

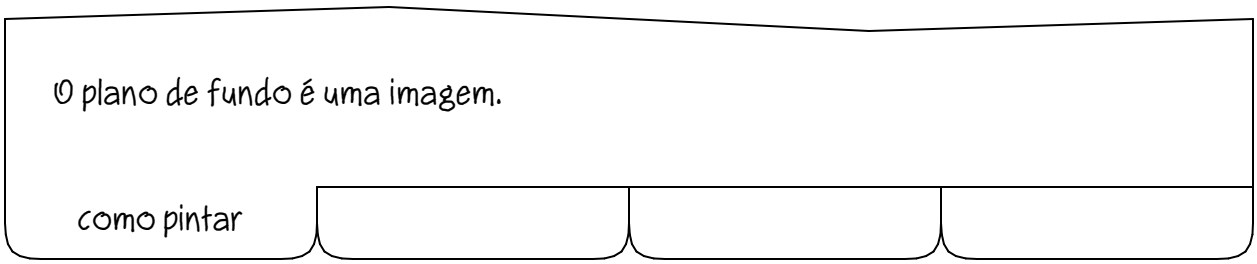
```
Para gerenciar o comando (atualizar):
Mostre tudo.

Para mostrar tudo:
Crie o painel de fundo.
Atualize a tela.
```

cOmo pintar

A razão pela qual criamos duas tarefas separados, uma para atualizar a tela e outra para mostrar tudo será explicada em breve. A tarefa de criar o painel do fundo será abordada nas próximas páginas.

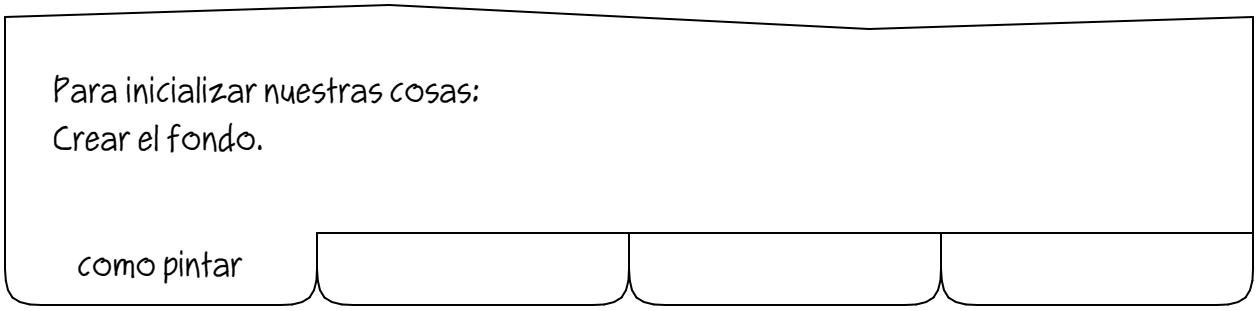
O FUNDO

Nosso painel de fundo começa com uma definição. Digite o seguinte:

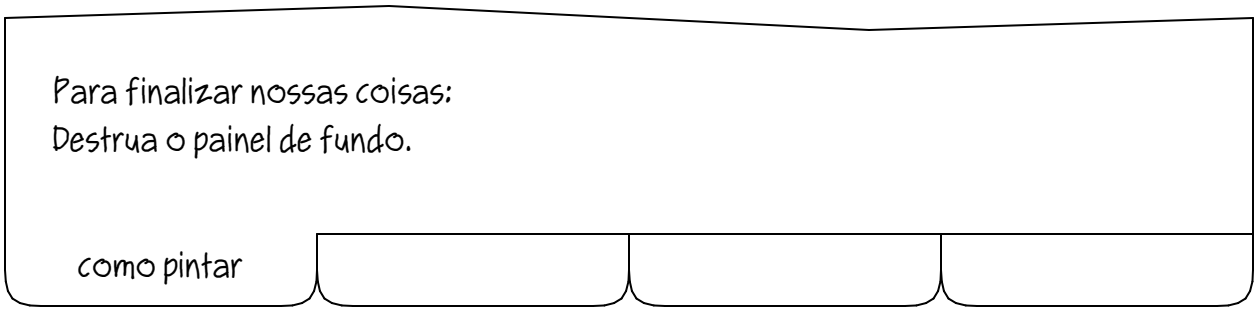


Se você cresceu programando em outras linguagens mais obscuras, provavelmente pensará no "painel de fundo" como uma variável global do tipo "imagem". Tudo bem. Sem problemas. Mas se você está apenas começando, você provavelmente vai pensar algo como "O fundo é uma imagem". E não há problema também. Não deixa de ser uma imagem, de certo modo.

Criaremos o painel de fundo quando inicializarmos nossas coisas:



E o destruiremos quando terminarmos:



VAZAMENTOS DE MEMÓRIA

O painel de fundo, como dissemos, é uma imagem. Imagens necessitam de memória para serem armazenadas. A quantidade necessária depende, evidentemente, do tamanho da imagem. Como nem sempre sabemos com antecedência quão grande ou pequena pode ser uma imagem, a memória para imagens é alocada, dinamicamente, no tempo de execução. Esta espaço ocupado de memória terá de ser desocupado mais tarde quando não for mais necessário.

Por convenção, utilizaremos as palavras chave criar e destruir sempre que efetuarmos alocação e desalocação de espaços de tamanho variável na memória. É de sua responsabilidade destruir tudo o que você criar antes de abdicar do controle em seu programa. Se você não fizer isso, você causará um "vazamento de memória" e pedaços de memória irão vazar de dentro do seu computador e cair nos seus sapatos.

Você será capaz de ver isso por si mesmo depois de criarmos nosso painel de fundo. Comente a linha que a destrói o painel, e quando você sair do programa usando CTRL-Q, uma caixa de mensagem assustadora aparecerá com as más notícias.

Se você já programou antes, provavelmente vai querer saber que...

(1) As strings são dinamicamente alocadas e podem ter qualquer comprimento — mas a memória é gerenciada inteiramente (e muito eficientemente) por mim, assim elas parecem estáticas para você. Em outras palavras, não se preocupe com elas. Apenas aproveite.

(2) Quando você destrói uma coisa, tudo que está ligado a essa coisa é destruído juntamente com ela. Isso libera você do tedioso fardo de escrever detalhadas destruições rotinas para cada tipo de coisa que você cria.

(3) Qualquer coisa mais do que isso cai na categoria "coleta de lixo" e, como cada programador maníaco sabe, a coleta de lixo é para garis.

Se você nunca programou, certifique-se de limpar o que você mesmo sujou.

CRIANDO, PINTANDO E PINCELANDO

Beleza, vamos voltar ao trabalho. Nós vamos criar o painel de fundo colorindo a tela usando tons de cinza, e atualizando a tela a cada 1000 milissegundos. Quando terminarmos, faremos uma cópia para que possamos usá-la durante os eventos de atualização.

Para que o plano de fundo seja criado:
Desenhe a borda da tela usando a cor branca nas bordas e a cor branca no interior.
Percorra.
Escolha um lugar aleatório em qualquer lugar dentro da tela.
Escolha uma cor aleatória variando entre a cor cinza bem clara e a cor branca.
Coloque a cor na posição escolhida anteriormente.
Se um contador tiver passado de 80000, pare.
Caso o contador seja igualmente divisível por 1000, atualize a tela.
Repita.
Copie o painel de fundo usando a borda da tela.
Para alterar a cor em um lugar:
Escolha a parte superior esquerda de uma elipse dentro de um raio de 2 milímetros da posição selecionada.
Escolha a parte inferior direita da elipse dentro de 1mm da posição selecionada.
Desenhe a elipse usando a cor da borda e a cor do interior.

como pintar

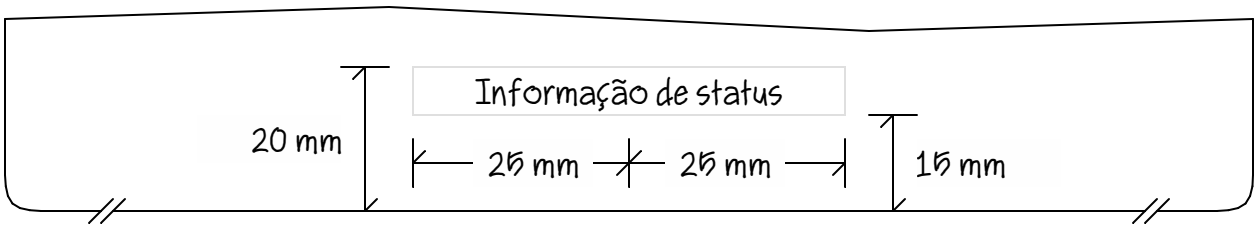
As cores são definidas no arquivo o cérebro.

A paleta de cores se encontra nas próximas páginas. Uma posição é um par de coordenadas x e y. Consulte a definição no o cérebro para saber mais. A elipse do pincelador simula inteligentemente a pintura da ponta de um pincel com graus de pressão variados.

Chega de falar. Vamos ver o que o nosso programa consegue fazer. Execute-o. Em seguida, dê um Alt-Tab algumas vezes para verificar se o comando de atualização de tela está sendo gerenciado corretamente pelo programa. Uau.

O PAINEL DE INFORMAÇÕES

De acordo com o nosso design, o programa Como Pintar deve exibir o status e as mensagens de erro na parte inferior central da tela. Mais ou menos assim:



E aqui estão as definições que precisamos para começar:

```
O painel de informações possui uma caixa e um texto.

Para inicializar o painel de informações:
Salve o centro da tela em uma posição.
Poner el izquierda del punto menos 25 mm en la izquierda del estado.
Poner el izquierda del punto más 25 mm en la derecha del estado.
Poner la inferior de la pantalla menos 20 mm en la superior del estado.
Poner la inferior de la pantalla menos 15 mm en la inferior del estado.

Para criar o painel de informações:
Dibujar la cadena del estado en la caja del estado (centrado).
```

como pintar

Nada de extraordinário aqui.

Mas note que apesar de termos criado o status, nós ainda não criamos a caixa — a gente só usou ela como base para posicionar corretamente o texto na tela.

O Canal de Comunicação (API) do Painel de Informações.

Agora vamos adicionar algum comandos triviais que, mesmo sendo triviais tornarão simples e fácil usar o nosso painel de informações. Aqui está a primeira rotina:

```
Para apagar o conteúdo do painel de informações:
Apague o texto do painel de informações.
Mostre tudo.

como pintar
```

Esta rotina será chamada no início de cada interação para garantir que as informações e mensagens de erro sejam limpas antes de se mostrar informações novas.

E aqui está a outra rotina:

```
Para exibir um texto no painel de informações:
Coloque o conteúdo do texto no texto do painel de informações.
Mostre tudo.

como pintar
```

Essa rotina será usada em todo o lugar para informar o usuário sobre o que estamos fazendo. Ele permite-nos definir a mensagem de status com uma única linha de código.

Se você é um programador experiente (e você não é um preguiçoso), você sabe quão úteis rotinas triviais como estas podem ser. Então não hesite em colocá-la no código do nosso programa. Se esta é sua primeira vez (ou você é um preguiçoso), escute o que estou dizendo.

Agora, duas das rotinas que criamos precisam ser atualizadas para poder fazer uso das nossas novas funcionalidades. Aqui estão elas já com o código novo no seu devido lugar:

```
Para mostrar tudo:
Crie o painel de fundo.
Crie o painel de informações.
Atualize a tela.

Para inicializar nossas coisas:
Crie o painel de fundo.
Inicialize o painel de informações.
Mostre "Olá, mundo!" no painel de informações.
```

como pintar

Faça as alterações nas duas rotinas, e então execute o nosso programa. Depois que o programa terminar de criar o painel de fundo, o programa exibirá a mensagem inicial no painel de informações, bem no centro inferior da tela, conforme imagem abaixo:

Ótimo. Já que a tela é praticamente refeita sempre que recebemos um comando de atualização, o painel de informações é mantido, mesmo se você apertar Alt+Tab. Experimente. Mais tarde, vamos ajustar a mensagem do painel de informações em vários lugares para refletir o estado atual do programa.

BOTÕES

A nossa mensagem de status era uma coisa única. Mas os nossos botões não são. Os seus nomes são diferentes, é claro, e cada um faz uso de uma rotina diferente. Mas sua forma e comportamento geral são idênticos.

Podemos, portanto, definir botão de uma forma genérica, junto com um punhado de de rotinas de suporte que funcionarão com qualquer botão. Começamos aqui:

Um botão tem uma caixa e um nome.

Para fazer um botão usando uma coordenada e um nome:
Coloque o valor da coordenada x no valor da coordenada esquerda do botão.
Subtraia a largura do nome do valor da coordenada esquerda do botão.
Coloque o valor da coordenada y no valor da coordenada do topo do botão.
Subtraia 5 milímetros do valor da coordenada do topo do botão.
Coloque o valor da coordenada no canto inferior direito do botão.
Coloque o nome informado no nome do botão.

como pintar

Se você é um programador inteligente com muita experiência e uma profunda compreensão de gramática, você será capaz de deduzir que o artigo indefinido no início da primeira definição indica que estamos definindo um tipo, não é uma variável.
Se você não for, você vai simplesmente pensar, "Um botão tem uma caixa e um nome". Também não há problemas em pensar dessa forma.

Mas se você é um leitor observador, experiente ou não, você irá concluir que botões não requerem alocação de memória dinâmica, já que usamos a palavra "fazer" em vez de "criar" no cabeçalho da segunda definição.

E você verá também, espero eu, que a largura de um botão depende do seu nome, e que o ponto com que começamos está na parte inferior direita do botão.

TRABALHANDO COM BOTÕES

Queremos ver nossos botões na tela, é claro, e queremos ser capazes de clicar neles para fazer as coisas acontecerem. Aqui estão algumas rotinas complementares:

```
Para desenhar um botão:
Desenhe o nome do botão na caixa do botão.

Para decidir se o mouse está em cima de um botão:
Se a localização do mouse está dentro da caixa do botão, diga sim.
Diga não.

como pintar
```

Como a caixa dos botões tem exatamente o tamanho certo para o nome que colocarmos nela, podemos colocar o nome no botão sem qualquer preocupação sobre alinhamento. Nós apenas colocamos o nome no botão na caixa do botão e pronto. Sinta-se livre para modificar a caixa do botão, mudando a margem ou espaçamento, bem como tamanho ou cor.

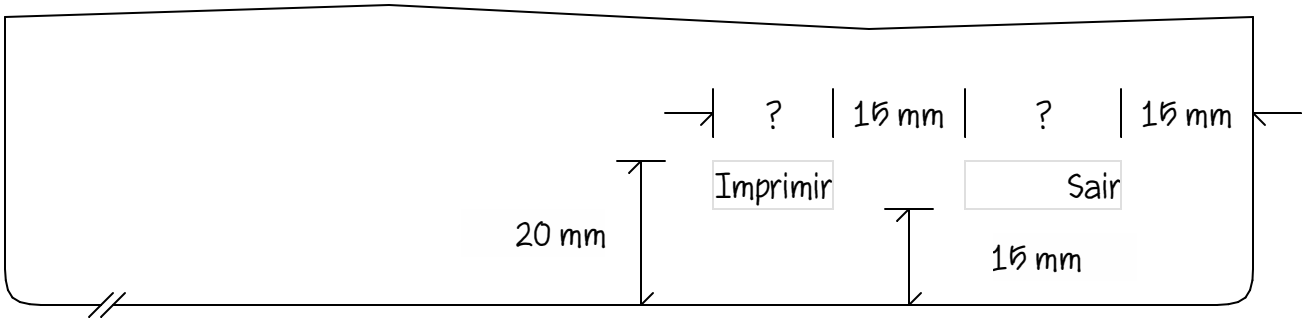
Brincadeirinha.

A segunda rotina é um exemplo de um tipo especial de rotina que me diz como tomar decisões. Rotinas desse tipo são chamadas de "decisoras", e elas sempre começam com as palavras Para decidir se. Se você teve o infortúnio de programar numa linguagem menos natural, você pode pensar em decisores como funções booleanas. É bem parecido.

Apenas lembre-se de que não existe meio termo numa função decisora. Para sair de um decisor, você deve chegar a uma conclusão definida usando os termos diga sim ou diga não. Nenhum outro comando que você utilize vai funcionar nesse caso.

HORA DE TRABALHAR COM OS NOSSOS BOTÕES

Esse aqui é o esboço dos botões do nosso projeto Como pintar:



E esse aqui é o código para criar os botões:

O botão de impressão é um botão.

O botão de sair é um botão.

Para inicializar os botões:
Coloque o valor da coordenada inferior da tela inferior numa coordenada y.
Subtraia 15 milímetros do valor da coordenada y.
Coloque o valor da coordenada direita da tela inferior numa coordenada x.
Subtraia 15 milímetros do valor da coordenada x.
Faça o botão de sair usando as coordenadas informadas e o termo "Sair".
Coloque o valor da coordenada esquerda ddo botão sair numa coordenada x.
Subtraia 15 milímetros do valor da coordenada x.
Faça o botão de impressão usando as coordenadas e o termo "Imprimir".
Faça o botão de impressão usando as coordenadas e o termo "Imprimir".

como pintar

Em primeiro lugar, definimos os nossos botões. Então posicionamos o botão Sair à distância de 1/2 polegada, a partir da extremidade direita da tela e 1/2 polegada acima a partir da extremidade inferior tela. Finalmente, usamos a extremidade esquerda do botão Sair — que foi calculado na rotina fazer um botão — para criar o botão de Impressão.

FAZENDO OS NOSSOS BOTÕES FUNCIONAREM

Quase lá. Precisamos atualizar o conteúdo de três das nossas rotinas e acrescentar mais uma no código do nosso programa, conforme demonstrado abaixo:

```
Para inicializar nossas coisas:
Crie o painel de fundo.
Inicialize o painel de informações.
Inicialize os botões.
Mostre "Olá, mundo!" no painel de informações.

Para mostrar tudo:
Crie o painel de fundo.
Crie o painel de informações.
Crie o botão de impressão.
Crie o botão de sair.
Atualize a tela.
Para manipular um evento (clique com o botão esquerdo):

Limpe o status.
Se o evento estiver no botão de impressão, imprima.
Se o botão de fechar for clicado, feche o programa.
Para imprimir:

Mostre "Imprimindo..." no painel de informações.
```

como pintar

Observe que consideramos cada clique como uma nova interação, a qual limpa conteúdo do painel de informações. E note que a rotina de impressão ainda não funciona de verdade, mas é já que a gente chega lá. Agora é hora de executar o programa para ver se não houve nenhum erro. Clique nos botões. Verifique o conteúdo do painel de informações. Clique no botão Imprimir. A mensagem Imprimindo... deverá ser exibida. Clique no botão Sair. O programa deve se fechar automaticamente.

O CAMPO DE TEXTO

Há uma componente bem útil chamada campo de texto no arquivo o cérebro que torna relativamente fácil criar campos de texto nos nossos programas. As funções Recortar, Copiar, Colar, Desfazer, Refazer, Selecionar e até mesmo Verificar Ortografia, são totalmente suportadas. Além de tudo, ele é um componente rápido e eficiente. O bloco de notas, por exemplo, é na verdade um super bloco de texto. Você pode aprender mais sobre o campo de texto no Glossário que fica no final deste livro.

No entanto, não vamos precisar implementar todas essas funções no nosso pequeno programa. Algo muito mais simples já é o suficiente. Aqui estão as definições básicas que nós vamos precisar para o nosso programa. Dê uma olhada com calma:

O campo de texto tem uma caixa e um texto.

Para inicializar o campo de texto:
Poner la izquierda de la pantalla más 15 mm en la izquierda del texto.
Poner la izquierda del texto más 50 mm en la derecha el texto.
Poner la parte inferior de la pantalla menos 20 mm en la superior del texto.
Poner la parte inferior de la pantalla menos 15 mm en la inferior del texto.

Para exibir o texto:
Poner la cadena del texto luego "_" en una cadena.
Coloque o texto na caixa do campo de texto.

como pintar

Nosso campo de texto é colocada 1/2 polegada acima da parte inferior esquerda da tela. Ele tem 1/4 de polegada de altura e 2 polegadas de largura. Eu não acho que precisemos de uma foto aqui.

Observe que, no entanto, que nós implementamos um cursor de texto bem básico, mas eficaz. Quando digitarmos o texto, um underline aparece depois do último caractere inserido.

FAZENDO O CAMPO DE TEXTO FUNCIONAR

NTemos de modificar duas das nossas rotinas para que o nosso campo de texto funcione. Também precisaremos gerenciar as atividades do teclado, mas lidaremos com isso nas próximas páginas. Por enquanto, apenas certifique-se de que você atualizou as seguintes rotinas::

Para inicializar nossas coisas:
Crie o painel de fundo.
Inicialize o painel de informações.
Inicialize os botões.
Inicialize o texto.
Mostre "Olá, mundo!" no painel de informações.

Para mostrar tudo:
Esconda o cursor do mouse.
Crie o painel de fundo.
Crie o painel de informações.
Crie o botão de impressão.
Crie o botão de sair.
Crie o campo de texto.
Atualize a tela.

como pintar

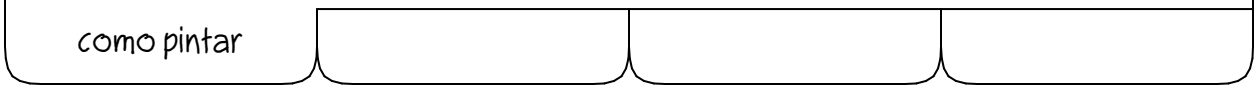
A única alteração na primeira rotina é a linha que inicializa o campo de texto.

A segunda rotina, porém, tem duas adições. Nós criamos o campo de texto, antes de atualizar a tela. Mas também ocultamos o cursor do mouse para que ele não atrapalhe o usuário quando ele estiver digitando. Mas não se preocupe com isso. O evento "exibir cursor" vai trazê-lo de volta sempre que o mouse se mover.

GERENCIANDO O PRESSIONAMENTO DE TECLAS

Consulte só como é que iremos modificar nosso geremciador de pressionamento de teclas para que ele encaminhe as teclas pressionadas:

Para gerenciar um comando (pressionamento de tecla):
Apague o conteúdo do painel de informações.
Se o comando for modificado, gerencie o comando (atalho); prossiga.
Se o valor da tecla for imprimível, gerencie o comando (imprimível); prossiga.
Salve a tecla do comando em uma tecla.
Se a tecla for a tecla Esc, gerencie o comando (cancelar); saia.
Se a tecla for a tecla backspace, gerencie o comando (apagar texto); prossiga.
Se a tecla for a tecla enter, gerencie o comando (tecla enter); prossiga.



Primeiramente, observe que consideramos qualquer tecla o início de uma nova interação, por isso nós limpamos qualquer mensagem anterior que estivesse no painel de informações. Talvez seja um pouco exagerado, mas pelo menos o painel de informações vai mostrar apenas o que interessa.

Em seguida nós gerenciamos as teclas de atalho da mesma forma que fizemos anteriormente.

Se o valor da tecla pressionada for uma letra, nós encaminhamos esse valor para outra rotina. Note que estamos verificando o valor do byte da tecla, e não a tecla em si. Temos que fazer isso porque uma tecla pode produzir caracteres imprimíveis, mas também caracteres não imprimíveis. Os comandos Ctrl+A e Alt+A, por exemplo, não são imprimíveis. Já o comando Shift-A é imprimível, mas tem um valor interno diferente do A e do a. Se você quer saber exatamente o que é um caractere imprimível, procure o termo no arquivo o cérebro.

Por último, colocamos a tecla que foi pressionada em uma variável local chamada tecla apenas deixar as três linhas que faltam um pouco mais curtas. Nós iremos adicionar mais quatro linhas uma para a tecla Home, outra para a tecla End, mais uma para a tecla Page Up e a última para a tecla Page Down.

GERENCIANDO (AINDA MAIS) PRESSIONAMENTOS DE TECLA

Aqui estão as rotinas que gerenciarão as teclas Home, End, Page Up, Page Down, e a tecla Enter:

```
Para gerenciar um comando (caractere imprimível):
Coloque a tecla digitada ao final do texto do campo de texto.
Mostre tudo.

Para gerenciar um comando (limpar texto):
Limpe a sequência de caracteres do campo de texto.
Mostre tudo.

Para lidar com um evento (apagar letra):
Se a o texto do campo de texto estiver em branco, alerte o usuário; prossiga.
Remova o último caractere do texto do campo de texto.
Mostre tudo.

Para gerenciar um comando (tecla Enter):
```

como pintar

Como mencionamos anteriormente, as letras, números e símbolos são simplesmente adicionadas ao texto. A tecla Esc limpa o conteúdo do campo de texto. O cursor de texto, no entanto, ainda aparecerá já que ele está anexado na rotina de criação mesmo que o campo de texto esteja em branco. A tecla ? irá apagar a última letra digitada, ou se não houver nenhum texto a ser apagado, irá emitir um som de alerta. Nas próximas páginas desenvolveremos a última rotina (a rotina da tecla Enter).

Agora vamos executar nosso programa e testá-lo um pouco. O cursor de texto está aparecendo? Excelente. Digite alguma coisa no campo de texto. O painel de informações apagou o Olá Mundo,? O cursor do mouse sumiu? Excelente. Agora dê um Alt+Tab. Continue até voltar ao Como pintar. Excelente. Teste a tecla ? para ver se o alerta sonoro funciona. Muito bem. Mova o mouse. A seta do mouse reapareceu? Ótimo. Clique no botão Sair. Ótimo.

A MÁGICA

Em nossa pequena caixa de texto, iremos digitamos o nome de qualquer pessoa, lugar ou coisa imaginável e apertar a tecla Enter. Após alguns segundos, uma pintura original, pintada no mesmo estilo de Claude Monet, aparecerá na tela. O usuário poderá ver várias outras obras de arte similares apenas utilizando as teclas Page Up e Page Down. Incrível.

Mas como é que vamos fazer com que isso aconteça?

É claro que se ele estivesse vivo, o próprio Claude Monet pintaria as obras de arte para nós. Como ele não está, nos simplesmente encontraremos alguas coisas que ficariam boas se ele pintasse. Em seguida, nós criaremos algumas obras de arte com base nesses coisas. E como criaremos uma obra de arte com base em um modelo? Mais uma vez, da mesma forma que o próprio Monet faria. Pegando a fotografia original, misturandp um pouco de tinta, e dando umas pinceladas por cima. Repetindo até terminar o quadro.

Tudo o que precisamos agora são (1) alguns modelos, e (2) uma rotina que pinte usando o modelo como base.

Bem, a segunda parte é relativamente fácil. Já ensinamos o Como pintar a dar uma pincelada na tela — é assim que nosso programa cria o plano de fundo toda vez que ele é executado. Dessa forma, a única diferença é que utilizaremos um modelo como base.

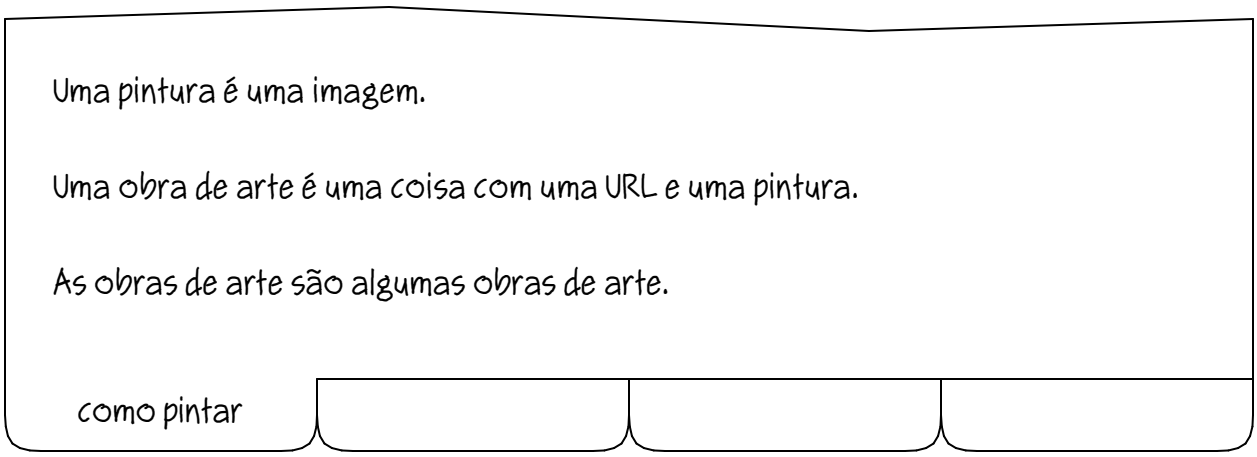
É a primeira parte que é complicada. Onde vamos encontrar imagens para qualquer coisa que o usuário digite? Uma paisagem desértica. Um arco íris. Uma Ferrari.
Não é muito prático guardar todo tipo de imagem em um programa que deveria ter poucos kilobytes. Ainda bem que isso não é necessário. Pra quê guardar imagens se temos o Google? No Google Imagens podemos encontrar qualquer imagem que pensarmos. De graça.

Eis o plano.

Quando a tecla Enter for pressionada, Google nos dará uma página cheia de URLs (localizadores de recursos) que contém as imagens relacionadas ao termo de pesquisa do usuário. Nós armazenaremos cada uma dessas URLs como se fosse um modelo esperando para virar obra de arte. Então, quando for hora de exibir a obra de arte, iremos pintá-la, usando a rotina que criamos.

OBRAS DE ARTE

Aqui estão as definições básicas de que precisamos para as nossas obras de arte:



A primeira linha faz de "pintura" um sinônimo de "imagem". Queremos deixar claro que nossas obras serão obras de arte originais, não apenas imagens baixadas do Google. Na verdade, você verá mais tarde que nós não iremos salvar as imagens que recebemos do Google. Uma vez que a obra de arte é criada, não precisamos mais da imagem..

Agora se você for especialista em estruturas de dados dinâmicos, preste atenção.

A palavra coisa na segunda definição é muito especial para mim. Ele indica um objeto de dados dinâmicos que pode ser associado a outros do mesmo tipo para formar uma cadeia - cada objeto apontando para o objeto anterior a ele, e para o objeto que estiver depois dele.
Se você está pensando que parece uma lista duplamente vinculada, você está certo. Rotinas para inserir, acrescentar e excluir coisas. podem ser encontradas no arquivo o cérebro.

As obras de arte, definidas na terceira linha, são um exempo dessas listas.

Agora, se você não é um especialista em estruturas de dados dinâmicos, você realmente não precisa entender todos esses detalhes. Só grave isso: "Uma obra de arte é uma coisa com uma URL e uma pintura" e "As obras de arte são algumas obras de arte".

A OBRA DE ARTE ATUAL

Cada URL que o Google nos dá se torna uma obra de arte — inicialmente elas são obras de arte "em andamento", que mais tarde são finalizadas e exibidas ao usuário. Precisamos criar uma forma de saber qual obra de arte está aparecendo na tela, e precisamos criar uma maneira de alternar entre diferentes obras de arte.

Aqui estão alguns códigos que irão nos ajudar nessa tarefa:

```
A obra de arte atual é uma obra de arte.

Para exibir uma obra de arte:
Se a obra de arte for inexistente, saia.
Mostre o termo "Trabalhando..." no painel de informações.
Defina a obra de arte como sendo a obra de arte atual.
Termine a obra de arte atual.
Limpe o conteúdo do painel de informações.
Mostre tudo.
```

como pintar

A obra de arte atual é uma referência à obra de arte que está sendo exibida atualmente na tela. Se não houver obras de arte sendo exibidas, não existirá nenhuma obra de arte atual. Essa situação ocorre quando o programa acaba de ser iniciado e também sempre que o Google não consegue encontrar nenhuma imagem. Quando uma nova solicitação é processada, a obra de arte atual é definida como a primeira obra de arte da lista. Mais tarde, ela será atualizada automaticamente toda vezque o usuário pressionar as teclas Page Up, Page Down, Home e End.

O método normal de definição da obra de arte atual é mostrado acima. Se a obra de arte solicitada for inexistente, nenhuma ação será tomada. A mensagem de status é necessária porque pode demorar um pouco pro nosso programa terminar de pintar a obra de arte. A mensagem é apagada, é claro, antes que o que a obra de arte seja exibida para o usuário.

TRABALHANDO COM OBRAS DE ARTE

Precisamos de uma rotina para pintar nossas obras de arte, é claro, mas (como qualquer artista) nós somente queremos exibir nossas obras que já estiverem prontas. O que significa que também precisamos de uma decissor para dizer qual é qual. Eis o código:

```
Para exibir uma obra de arte:
Se a obra de arte for inexistente, saia.
Se a obra de arte não estiver pronta, saia.
Crie a obra de arte.

Para decidir se a obra está terminada:
Se a obra de arte não existir, diga sim.
Se a obra de arte existir, diga sim.
[Caso contrário] diga não.
```

como pintar

A primeira rotina verifica se a obra de arte está pronta para ser pintada, e se estiver, ela pinta a obra de arte. Nós ainda não criamos uma rotina chamada crie uma pintura, e advinha só, nós também não vamos criar uma. Mas já que dissemos que uma pintura é uma figura, nós podemos simplesmente usar a rotina padrão criar uma imagem que está presente no arquivo o cérebro para fazer esse trabalho.

Se você é um nerd, você vai reconhecer isso como uma "redução automática de tipo" e vai me perguntar como que o compilador faz isso com tanta eficiência e elegância. Se você não é um nerd, você provavelmente vai pensar: "Tá, mas e daí?" e depois se perguntar por que as outras linguagens de programação não tem essa capacidade.

A segunda rotina é apenas um decisor comum. Note, no entanto, que ela considera uma obra de arte inexistente como se ela já estivesse finalizada. Até porque, se não houver trabalho pra fazer, isso significa que o trabalho acabou, certo?

FAZENDO NOSSAS OBRAS DE ARTE FUNCIONAREM

Há duas atualizações que temos que colocar no nosso código. A primeira alteração que precisamos fazer está na rotina de exibição. Felizmente não vamos mais precisar alterar ela. O código completo é parecido com este:

```
Para mostrar tudo:
Esconda o cursor do mouse.
Crie o painel de fundo.
Crie o painel de informações.
Crie o botão de impressão.
Crie o botão de sair.
Crie o campo de texto.
Crie a obra de arte atual.
Atualize a tela.
```

como pintar

Note que já que nós só atualizamos a tela no final da rotina, os componentes do programa podem ser criados em qualquer ordem. Exceto pelo painel de fundo, é claro.

Uma obra de arte é uma coisa, e as coisas são sempre alocadas dinamicamente, então precisamos limpá-las quando terminarmos. Aqui está a última rotina de finalização:

```
Para finalizar nossas coisas:
Destrua o painel de fundo.
Destrua as obras de arte.
```

como pintar

Agora vamos começar a trabalhar na tecla Enter. Eis o código:

```
Para gerrenciarmos um comando (tecla Enter):
Se a o texto do campo de texto estiver em branco, alerte o usuário; prossiga.
Mostre o termo "Trabalhando..." no painel de informações.
Coloque "http://images.google.com/images?q=" em uma URL.
Converta o texto do campo de texto em um texto de pesquisa.
Coloque o texto de pesquisa no fim da URL.
Leia a URL em um buffer.
Se o erro de entrada/saída não estiver em branco, mostre o erro no painel de
informações; saia.
Crie as obras de arte usando o buffer.
Se as obras estiverem vazias, mostre "Hã?" no painel de informações, saia.
Vá para a primeira obra de arte.

como pintar
```

Se o campo de texto estiver em branco, não há nada a fazer; nós usaremos o som para alertar o usuário e sairemos.

Caso contrário, nós vamos escrever uma mensagem no painel de informações (caso o Google esteja ocupado e não responda imediatamente). Em seguida, formulamos uma solicitação usando um texto literal e uma versão do texto que seja compatível com o formato HTML, lendo a resposta em um buffer (que é apenas um nome chique para uma sequência de caracteres).

Se algo deu errado, relatamos o erro. Se a página chegar intacta, tentamos gerar nossas obras de arte em andamento a partir dos dados no buffer. Se as obras estiverem em branco quando terminarmos, isso significa que o Google não entendeu nossa consulta — neste caso, dizemos "Hã?" e saímos. Caso contrário, mostramos ao usuário a primeira obra de arte da lista.

PERCORREDORES

Antes de continuarmos com nosso programa, preciso tirar um momento e falar com você sobre análise sintática. Análise sintática é a arte de percorrer um grande bloco de texto um palavra por vez, onde essa palavra pode ser tão pequena quanto uma letra ou tão grande quanto o bloco inteiro. Vamos usar essa sequência de palavras abaixo como nosso exemplo de bloco de texto:

"OLÁ DOUTOR NOME CONTINUAR ONTEM AMANHÃ"

E digamos que queremos extrair cada uma dessas palavras de forma individual. As ferramentas que usaríamos são (1) o subtexto e (2) o percorredor.

Um "subpalavra" é definida no arquivo o cérebro como um subconjunto do tipo "texto", que tem dois ponteiros de byte chamados primeiro e último. E quando você "colocar uma subpalavra" no nosso texto de amostra, o compilador faz com que o primeiro byte aponte para o O na palavra OLÁ e o último byte aponte para Ã na palavra AMANHÃ.

No entanto, você será capaz de encontrar o "percorredor" no o cérebro. Ele consiste de três subtextos: um subtexto original, um subtexto fonte e um token. E quando você "colocar um percorredor" no nosso texto de amostra, o compilador pega o subtexto original e o subteto fonte no texto (como acima) e define o token como vazio. Então quando você usa o comando "mova o percorredor (regras de amostra)", o compilador aponta a letra D da palavra DOUTOR como primeiro byte do subtexto fonte. A letra O da palavra OLÁ é considerada como o primeiro byte do token, e a letra Á de OLÁ é considerada como o último byte do token. Quando você der novamente o comando, o percorredor se moverá para a próxima palara, sendo que a os bytes do token ficarão na palavra DOUTOR e a fonte ficará na letra N da palavra NOME.
Pegou a ideia? Excelente.

Agora aqui está a parte realmente estranha: você pode usar o precorredor para criar suas próprias rotinas e extrair qualquer tipo de dado de qualquer tipo de arquivo. Mover um percorredor (regras do compilador), por exemplo, é a rotina que uso para analisar o código fonte do programa. Mova um percorredor (regras de verificação ortográfica) é utilizado para verificar a ortografia. Em breve, vamos programar a rotina Mova um percorredor (Google Imagens) para analisar os dados que vamos pegar da internet.

OBRAS DE ARTE EM ANDAMENTO

Aqui está o código para criar nossas obras de arte em andamento a partir dos dados do Google:

```
Para criar algumas obras de arte usando um buffer:
Destrua as obras de arte.
Limpe a obra de arte atual.
Coloque um percorredor no buffer.
Percorra.
Mova o percorredor (usando o Google Imagens).
Se o token do percorredor estiver em branco, saia.
Crie uma obra de arte usando o token do percorredor.
Acrescente a obra de arte à lista de obras de arte.
Repita.

Para criar uma obra de arte a partir de uma URL:
Reserve memória para a obra de arte.
Defina a URL como a URL da obra de arte.
```

como pintar

Nós nos livramos de qualquer obra de arte antiga e redefinimos o trabalho atual para que não aponte mais para a obra de arte que acabamos de destruir. Então criamos um percorredor e inserimos um laço de repetição.

Dentro do laço, nós movemos o percorredor para a próxima imagem da página. Se não houver mais imagens, nós fizemos algo histórico (afinal de contas, para cada termo existem centenas de correspondências). Se houver, criamos um trabalho em andamento com o comando crie uma obra de arte usando uma URL. Mesmo que o token do percorredor não seja uma URL, eu sei que uma URL é apenas um texto, e que o token do percorredor é um subtexto. Como nenhuma outra rotina cria uma obra de arte usando um texto, o compilador acaba por chamar a rotina correta. Em seguida, anexamos a obra de arte na lista de obras e repetimos o processo.

MOVENDO OS NOSSOS PERCORREDORES

Aqui estão as rotinas que precisamos para percorrer as imagens do Google:

```
Para mover um percorredor (usando o Google Imagens):
Limpe o token do percorredor.
Percorra.
Se o token do percorredor estiver em branco, saia.
Se a fonte do piloto percorredor começar com "src=""http://t", pare.
Adicione 1 ao primeiro byte da fonte do percorredor.
Repita.
Adicione o comprimento do "src=""" ao primeiro byte da fonte do percorredor.
Posicione o token do percorredor na fonte do cavaleiro.
Movao percorredor (usando regras de atributos HTML).

Para mover um percorredor (usando regras de atributos HTML):
Se a fonte do percorredor estiver em branco, saia.
Se o alvo do primeiro byte da fonte do percorredor for sinal de maior, saia.
Se o alvo do primeiro byte da fonte do percorredor for uma aspa dupla, saia.
Avance o percorredor.
Repita.
```

como pintar

Caso você queira ver os dados retornados pelo Google, você pode salvar o conteúdo da fonte do percorredor em um arquivo usando a rotina gravar um buffer num arquivo que está presente no arquivo o cérebro.

Observe que, uma vez que as subpalavras contêm ponteiros de byte, não os bytes em si, você precisa especificar o alvo do primeiro byte da fonte do percorredor para obter os dados. Eu percebo que isto é um pouco complicado, mas sempre que a gente lida com coisas complicadas, é isso que ocorre.

PREPARANDO A PINTURA

Estamos quase prontos para terminar uma obra de arte. Mas antes que isso aconteça, vamos programar algumas rotinas auxiliares para facilitar as coisas para nós. Aqui estão elas:

```
Para escolher um lugar perto de uma caixa:
Privatize a caixa.
Recue a caixa 3 milímetros à direita.
Escolha uma posição aleatória em qualquer lugar dentro da caixa.

Para misturar uma cor usando uma posição como ponto de partida:
Obtenha a cor usando a posição informada.
Se a cor não for muitíssimo clara, prossiga.
Escolha uma cor aleatória variando entre a cor cinza bem clara e a cor branca.
```


A primeira rotina escolhe um lugar em qualquer lugar — ou perto de — uma caixa. Isso nos permite pincelar as bordas de nossa imagem de uma forma muito artística. Isso também nos permite misturar algumas das cores de fundo, assim o contraste entre a pintura e o fundo não fica tão gritante.

A declaração "privatize", caso esteja se perguntando, cria uma cópia da caixa para que possa mudá-la sem afetar sem querer a rotina que chamou a função. A cópia da caixa original recebe o nome "caixa"; a caixa original recebe o nome "caixa original".

A segunda rotina é nossa rotina de pincelar. Ela obtém uma cor do modelo e dá uma pincelada — a menos que a cor seja quase branca, nesse caso nós substituímos pela cor de fundo. Isso dá a nossas pinturas um toque de "transparência" que aumenta consideravelmente sua atratividade.

PINTANDO

Ah, se o Monet pudesse nos ver agora! Vamos direto ao assunto:

```
Para finalizar uma obra de arte:
Se a obra de arte for inexistente, saia.
Se a obra de arte estiver pronta, saia.
Crie uma imagem usando a URL da obra de arte.
Se a imagem estiver vazia, saia.
Redimensione a imagem para 5 polegadas de largura por 5 polegadas de altura.
Centralize a imagem na caixa da tela.
Crie o painel de fundo.
Crie a imagem.
Percorra.
Escolha uma posição aleatório em qualquer perto da borda da imagem.
Misture uma cor usando a posição escolhida.
Coloque a cor na posição escolhida anteriormente.
Se um contador tiver passado de 20000, pare.
Repita.
Extraia a pintura da obra de arte usando o conteúdo da imagem.
Destrua a imagem.
```

como pintar

Se a obra de arte estiver em branco ou já estiver concluída, nós a ignoramos. Caso contrário, buscamos o modelo da internet, redimensionamos, centralizamos, e desenhamos o modelo em um fundo novo. Aí sim a gente começa a picelar. Muito. Quando terminarmos, nós extraimos a pintura da tela do quadro. Já que não vamos mais precisar do modelo, nós destruímos ele.

Vá em frente. Experimente. Gostou?

NAVEGANDO PELAS OBRAS DE ARTE

Aposto que você gostaria de ver todas as pinturas de cada termo pesquisado. Eu sei que sim. Para isso, precisaremos modificar o nosso gerenciador de pressionamento de teclas, e também será preciso adicionar quatro rotinas auxiliares. Aqui está a versão final do nosso gerenciador:

```
Para gerenciar um comando (pressionamento de tecla):
Apague o conteúdo do painel de informações.
Se o comando for modificado, gerencie o comando (atalho); prossiga.
Se o valor da tecla for imprimível, gerencie o comando (imprimível); prossiga.
Salve a tecla do comando em uma tecla.
Se a tecla do comando for a tecla Esc, gerencie o comando (cancelar); saia.
Se a tecla for a tecla Backspace, gerencie o comando (apagar texto); prossiga.
Se a tecla for a tecla Enter, gerencie o comando (tecla enter); prossiga.
Se a tecla for a tecla Home, gerencie o comando (tecla home); prossiga.
Se a tecla for a tecla End, gerencie o comando (tecla end); prossiga.
Se a tecla for a tecla Page Up, gerencie o comando (tecla page up); prossiga.
Se a tecla for a tecla Page Down, gerencie o comando (tecla page down); prossiga.

como pintar
```

A tecla Home nos mostra a primeira obra de arte da lista. Se a primeira obra de arte da lista já estiver sendo exibida, nós iremos alertar o usuário.

A tecla End nos mostra a última obra de arte da lista. Se a última obra de arte da lista já estiver sendo exibida, nós iremos alertar o usuário.

A tecla Page Up mostra a obra de arte que estiver antes da obra de arte atual. Se a primeira obra de arte da lista estiver sendo exibida, nós iremos alertar o usuário.

A tecla Page Down nos mostra a próxima obra de arte da lista. Novamente, se não houver mais nenhuma obra de arte na lista, nós alertaremos o usuário.

Home, End, Page Up, & Page Down

Aqui estão as rotinas auxiliares de que precisamos para navegar entre as obras de arte. Dê uma olhada com calma:

```
Para gerrenciarmos um comando (tecla Home):
Se a obra de arte estiver em branco, alerte o usuário; prossiga.
Se a obra de arte atual é a primeira obra de arte, alerte o usuário, prossiga.
Vá para a primeira obra de arte.

Para gerrenciarmos um comando (tecla End):
Se a obra de arte estiver em branco, alerte o usuário; prossiga.
Se a obra de arte atual é a última obra de arte, alerte o usuário, prossiga.
Vá para a última obra de arte.

Para gerrenciarmos um comando (tecla Page Down):
Se a obra de arte estiver em branco, alerte o usuário; prossiga.
Se a próxima obra de arte estiver em branco, alerte o usuário; saia.
Vá para a próxima obra de arte.

Para gerenciarmos um comando (tecla Page Up):
Se a obra de arte estiver em branco, alerte o usuário; prossiga.
Se a obra de arte anterior estiver em branco, alerte o usuário; saia.
Vá para a obra de arte anterior.
```

como pintar

A resposta será mais lenta na primeira vez que você criar uma obra de arte, já que temos que pintá-la antes de mostrá-la. A mensagem "Trabalhando..." aparecerá no painel de informações.

Experimente. Com certeza você vai gostar.

IMPRIMINDO

Bem, não há mais nada para fazer a não ser atualizar nossas rotinas de impressão:

```
Para imprimir uma obra de arte:
Se a obra de arte estiver em branco, alerte o usuário; saia.
Mostre o termo "Imprimindo..." no painel de informações.
Comece a impressão.
Crie uma página usando uma orientação horizontal.
Defina o fundo da folha.
Centralize a pintura da obra de arte atual na folha.
Transfira a pintura da obra de arte atual para a folha.
Centralize a pintura da obra de arte atual na tela.
Finalize a página.
Termine de imprimir.
Mostre o termo "Obra de Arte Impressa com sucesso" no painel de informações.
```

como pintar

Nós iremos ajustar a posição da pintura para que ela possa ser impressa bem no meio da folha, a partir daí nós efetivamente colocamos a pintura na folha, e pronto.

Já colocamos o botão de impressão na o lugar certo, mas não criamos a rotina com o comando de imprimir. Faça com que seu gerenciador de impressão fique parecido com isto:

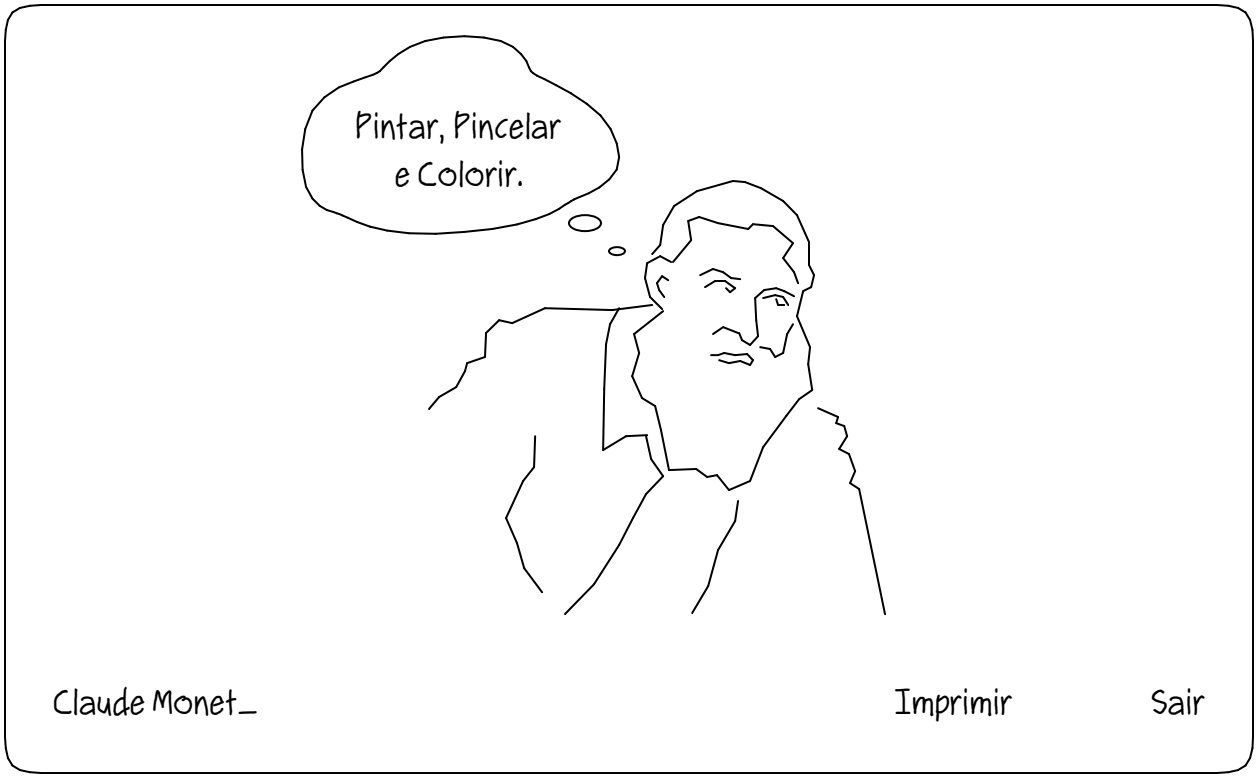
```
Para gerenciar um comando (atalho):
Se a tecla do comando for a tecla p, imprima a obra de arte; saia.
Se a tecla do comando for a tecla q, feche o programa; saia.
```

como pintar

UM ÚLTIMO AVISO

Aí está. Nosso programa está pronto. Um incrível aplicativo feito totalmente em português que "pinta" imagens de quase qualquer pessoa, lugar, ou coisa no estilo "inigualável" de Claude Monet. Tudo isso usando menos de 300 linhas de código.

Aqui está uma coisinha para nos lembrarmos.



PS: Não se esqueça de testar o programa buscando coisas como "nascer do sol". Ou outras paisagens. O fundo do mar. Montanhas. Rios. Flores e árvores. Pássaros e abelhas. Todas as criaturas, grandes e pequenas. Pessoas famosas. Pessoas comuns. Trens, barcos, aviões, carros antigos. Londres, Paris, Washington DC. Presentes.

A partir de agora, use o Glossário a seguir para guiá-lo em sua jornada na criação de seus próprios programas. Em Português Puro e Simples.

Glossário Explicativo de Termos e Definições

VISÃO GERAL

As sessenta páginas a seguir podem ser consideradas como um espécie de atlas alfabético do meu compilador e do cérebro. Se você fez o seu dever de casa (o nosso programa de exemplo), você deve ser capaz de lê-lo do início ao fim e saber do que estou falando. Mas vamos revisar, por precaução:

Espero que seus programas consistam em arquivos de texto armazenados em uma única pasta. Um desses arquivos deve ser uma cópia do o cérebro. Não importa em qual ordem os os arquivos estão. O nome do arquivo não importa, desde que o arquivo não possua nenhuma extensão, isto é, nenhum ponto.

Você pode invoca o compilador dentro do bloco de notas (o editor de arquivos). Apenas abra qualquer arquivo de texto dentro da pasta que você deseje compilar e use o comando Executar. Para encerrar um programa problemático, dê um Alt+Tab volte para a tela do compilador e use o comando Parar.

O arquivo executável será salvo na mesma pasta do arquivo que foi compilado. O nome do arquivo executável será o mesmo nome da pasta em que ele estiver. Você pode renomear, modificar e distribuir seus arquivos executáveis do jeito que você quiser. Eles são livres de direitos autorais e não precisam de dll's para serem executados.

Os arquivos podem conter comentários e três tipos de definições: tipos, variáveis globais e rotinas. Em qualquer ordem que você achar melhor. Você pode usar letras maiúsculas, minúsculas, misturar tudo, tanto faz. As rotinas podem conter dois tipos de comandos: CONDICIONAIS e IMPERATIVOS. Se você está ficando confuso tentando ler esta seção do começo ao fim, tente pesquisar os tópicos acima no restante do documento, e aí sim volte para o glossário.

Um aviso importante. O compilador não permite o uso de comandos condicionais aninhados. (Ou seja, um Se dentro de outro). O compilador não permite o uso de comandos de repetição aninhados. (Ou seja, um loop dentro de outro). O compilador também não trabalha com números reais, equações, ou qualquer outro tipo de número que não seja inteiro ou racional. Basicamente o objetivo é utilizar a menor quantidade possível de matemática nos programas, assim como você faz na vida real.

ARQUIVOS

O sistema de arquivos do Windows é uma coisa de beleza insuperável, se igualando a uma obra de arte... Brincadeirinha. Na verdade o sistema de arquivos do Windows é uma verdadeira bagunça. Só pra você ter uma idéia:

Um endereço completo é um texto.	\ nome completo do arquivo = c:\pasta1\arquivo.exe
Um endereço de unidade é um texto.	\ Exemplo -> c:\
Um endereço de pasta é um endereço completo.	\ Exemplo -> c:\pasta1\pasta2\
Um nome de pasta é um texto.	\ apenas o nome da pasta + barra invertida = pasta2\
Um nome completo de arquivo é um texto.	\ desginador = etensão. Exemplo: aquivo.exe
Uma extensão de arquivo com ponto é um texto.	\ Exemplo: .ext
Um designador é um texto.	\ nome da pasta ou nome do arquivo

Mesmo com tanta cofusão, o compilador consegue:

EXTRAIR qualquer uma dos itens acima USANDO um endereço completo.

Ou ainda:

GRAVAR um endereço completo NO SISTEMA DE ARQUIVOS.
SUBSTITUIR um endereço completo POR outro endereço completo NO SISTEMA DE ARQUIVOS.
EXCLUIR um endereço completo NO SISTEMA DE ARQUIVOS.
CRIAR UMA CÓPIA DE um endereço completo EM outro endereço completo NO SISTEMA DE ARQUIVOS.

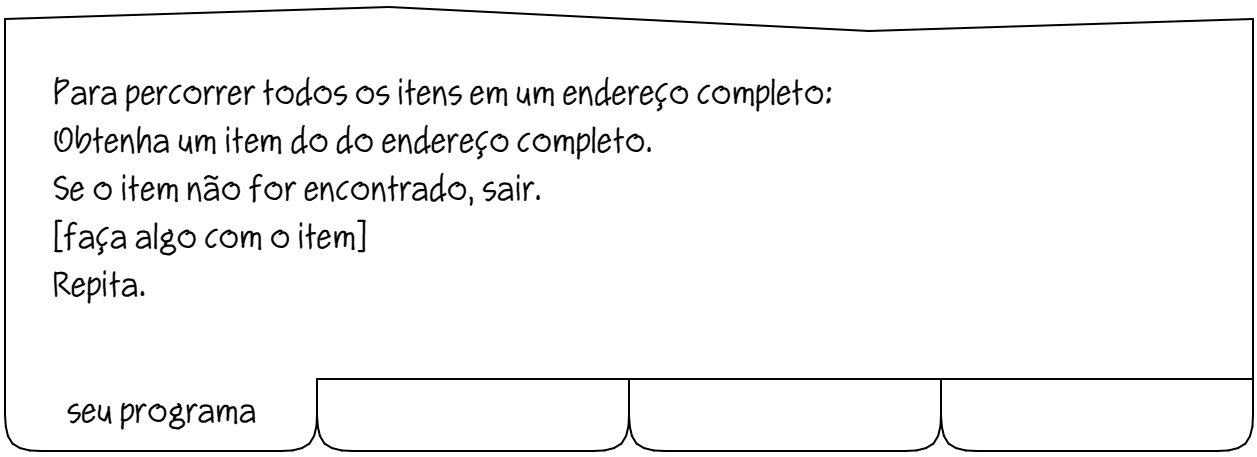
Ou, se você preferir:

COLOCAR um endereço completo DENTRO DE um texto.
GRAVAR um texto EM um endereço completo.

Se algo der errado, o erro de entrada/saída irá conter uma descrição do problema, adequada para exibição ao usuário. Você não precisa limpar o erro de entrada/saída antes de uma invocar uma dessas funções, mas deve verifica o erro depois de chamar a função para ter certeza que ele está em branco.

ARQUIVOS (continuação)

Se você precisar acessar uma ou mais pastas do sistema de arquivos, você pode usar um comando como esse:



O termo "Item" é definido da seguinte forma no compilador:

Um item tem
uma categoria,
um endereço completo, um endereço de pasta, um designador, uma extensão,
um tamanho,
um win32finddata e um número identificador.

O campo categoria é uma variável do tipo texto. O conteúdo dessa variável pode ser pasta ou arquivo para cada item encontrado. Os campos extensão e tamanho serão preenchidos apenas se a categoria for arquivo. Os campos win32finddata e número identificador são necessários que o compilador funcione no Windows.

Você também pode usar os seguintes comandos:

OBTENHA uma contagem de ITENS EM um endereço completo no SISTEMA DE ARQUIVOS.
OBTENHA um tamanho USANDO um endereço completo no SISTEMA DE ARQUIVOS.

Note que os contadores e tamanhos de arquivo, incluindo o tamanho no registro de nome Item, são limitados a 2147483647, que é o maior número permitido pelo compilador no momento.

ARITMÉTICA

O compillador tabalha com aritmética básica. O compilador mantém um registo do que está no arquivo o cérebro. Se possível, dê uma olhada com calma no arquivo. É um arquivo longo mas vale a pena. Para resumir a história, a essência e o poder do compilador reside no fato dele ser capaz de entender declarações do tipo:

```
ADICIONE isso Nisto.
SUBTRAIA isso Daquilo.
MULTIPLIQUE isso POR aquilo.
DIVIDA isso POR aquilo.
```

Caso você precise fazer uma divisão com resto, você pode utilizar o comando:

```
DIVIDA isto POR aquilo RETORNANDO o quociente E o resto.
```

Além disso, o compilador é capaz de:

```
ARREDONDAR algum número/variável PARA CIMA ATÉ ATINGIR O MÚLTIPLO MAIS PRÓXIMO DE outro número/variável.

ARREDONDAR algum número/variável PARA BAIXO ATÉ ATINGIR O MÚLTIPLO MAIS PRÓXIMO DE outro número/variável.
```

O compilador também é capaz de:

```
REMOVER O SINAL de um número/variável (extrair o módulo do número).
INVERTER O SINAL de um número/variável.
```

O compilador é capaz de até mesmo de:

```
REDUZIR uma fração (razão/proporção).
```

Você pode usar mais de uma operação aritmética no mesmo comando, usando as seguintes palavras chave: MAIS, MENOS, VEZES e DIVIDIDO POR. Você pode ler mais sobre esses operadores na seção Expressões Aritméticas deste glossário.

ASCII

Este é o Código Padrão Americano para o Intercâmbio de Informação (Extended ASCII) extendido, mais conhecido como Windows-1252 ou CP-1252. Devido a facilidade de implementação e ao fato do código ASCII cobrir a maior parte dos caracteres da língua portuguesa, ele foi escolhido comoo padrão para converter bytes em caracteres legíveis. Não é a melhor opção, já que existe o Unicode mas é uma das codificações de texto mais amplamente aceitas do planeta.

000 NUL	001 SOH	002 STX	003 ETX	004 EOT	005 ENQ	006 ACK	007 BEL	008 BS	009 HT	010 LF	011 BT	012 FF	013 CR	014 SO	015 SI
016 DLE	017 DC1	018 DC2	019 DC3	020 DC4	021 NAK	022 SYN	023 ETB	024 CAN	025 EM	026 SUB	027 ESC	028 FS	029 GS	030 RS	031 US
032	033 !	034 "	035 #	036 $	037 %	038 &	039 '	040 (	041)	042 *	043 +	044 ,	045 -	046 .	047 /
048 0	049 1	050 2	051 3	052 4	053 5	054 6	055 7	056 8	057 9	058 :	059 ;	060 <	061 =	062 >	063 ?
064 @	065 A	066 B	067 C	068 D	069 E	070 F	071 G	072 H	073 I	074 J	075 K	076 L	077 M	078 N	079 O
080 P	081 Q	082 R	083 S	084 T	085 U	086 V	087 W	088 X	089 Y	090 Z	091 [	092 \	093]	094 ^	095 _
096 `	097 a	098 b	099 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o
112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z	123 {	124 |	125 }	126 ~	127 DEL
128 €	129	130 ,	131 ƒ	132 „	133 …	134 †	135 ‡	136 ˆ	137 ‰	138 Š	139 ‹	140 Œ	141	142 Ž	143
144	145 ‘	146 ’	147 “	148 ”	149 •	150 –	151 —	152 ˜	153 ™	154 š	155 ›	156 œ	157	158 ž	159 Ÿ
160	161 ¡	162 ¢	163 £	164 ¤	165 ¥	166 ¦	167 §	168 ¨	169 ©	170 ª	171 «	172 ¬	173 -	174 ®	175 ¯
176 °	177 ±	178 ²	179 ³	180 ´	181 µ	182 ¶	183 ·	184 ¸	185 ¹	186 º	187 »	188 ¼	189 ½	190 ¾	191 ¿
192 À	193 Á	194 Â	195 Ã	196 Ä	197 Å	198 Æ	199 Ç	200 È	201 É	202 Ê	203 Ë	204 Ì	205 Í	206 Î	207 Ï
208 Ð	209 Ñ	210 Ò	211 Ó	212 Ô	213 Õ	214 Ö	215 ×	216 Ø	217 Ù	218 Ú	219 Û	220 Ü	221 Ý	222 Þ	223 ß
224 à	225 á	226 â	227 ã	228 ä	229 å	230 æ	231 ç	232 è	233 é	234 ê	235 ë	236 ì	237 í	238 î	239 ï
240 ð	241 ñ	242 ò	243 ó	244 ô	245 õ	246 ö	247 ÷	248 ø	249 ù	250 ú	251 û	252 ü	253 ý	254 þ	255 ÿ

O compilador possui variáveis globais com nomes como o byte de vírgula para cada elemento da tabela, assim você não precise trabalhar diretamente com os números deles. Você pode encontrar todas essas variáveis procurando a frase é um byte igual à no arquivo o cérebro.

BITS

Um "bit", como definido no arquivo o cérebro, é uma unidade de medida. É usado em comandos como 1 bit ou alguns bits. Você provavelmente não vai precisar dele a menos que você seja um ned que goste de passar o tempo manipulando bits usando comandos como:

CONJUNCIONE este valor COM este valor.(AND)
DISJUNCIONE este valor COM este valor.(OR)
DISJUNCIONE EXCLUSIVAMENTE este valor COM este valor.(XOR)

Em cada um destes casos, o primeiro argumento informado é o que é modificado pelo conectivo lógico.

Também é possível efetuar as seguintes operações lógicas:

DESLOQUE este valor alguns bits PARA A ESQUERDA.
DESLOQUE este valor alguns bits PARA A DIREITA.

Ou ainda:

SEPARE este valor EM um valor E em outro valor.

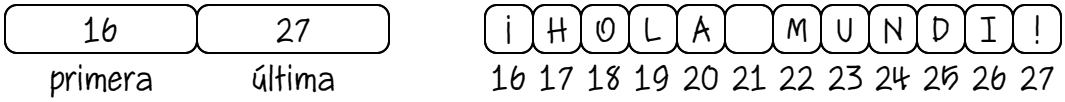
Como por exemplo, um byte em pedaços de 4 bits, ou 4 bits em pedaços contendo dois bytes, etc. Creio que raramente você precisará fazer isso.

Agora, se você não tem mínima idéia do que estou falando aqui, você não é um nerd e não deve se preocupar com isso. Você provavelmente não vai querer saber.

Mas se você entender o que eu estou dizendo, tenho certeza que você também vai curtir o tópico Windows bem no final, e a parte sobre literais nibles dentro do tópico Literais. Sem mencionar a questão do Possessivo, e os Imperativos. Além de todas as rotinas de baixo nível do compilador que usam a instrução INTEL e/ou o registro EAX.

CADENAS

Almaceno "cadenas" en dos partes: un registro incorporado con un par de punteros de bytes llamados primera y última (subcadena), y una matriz dinámica que contiene los bytes reales, como así: as. Puede copiar estas rutinas y hacer una decantación.



Los números en el diagrama, en caso de que no lo hayas adivinado, son direcciones ficticias. Una cadena está en blanco si la primera es nula (aún no hay memoria asignada), o la última es menor que la primera (lo que me permite preasignar la memoria). Tenga en cuenta que, aunque la parte de datos de una cadena se asigna dinámicamente, nunca tendrá que "crear" o "destruir" cadenas. Me ocupo de todo para que puedas:

PONER algo EN una cuerda.
ADJUNTAR una cadena A otra cadena.
ELIMINAR EL PRIMER BYTE DE una cadena.
ELIMINAR EL ÚLTIMO BYTE EN un cadena.

Además, puede concatenar cadenas con cadenas y otros tipos de datos utilizando el operador infijo LUEGO. Consulte el tema sobre "Expresiones" para obtener una descripción de la forma inteligente en que mis creadores implementaron esto.

CAIXAS

Um dos aspectos chave do compilador é a forma que ele trabalha com caixas. Uma caixa é uma variável do tipo record. O código da caixa é basicamente assim:

Um lado é uma coordenada.

Uma caixa tem
Um lado esquerdo,
Um lado de cima,
Um lado direito,
Um lado de baixo,
Um canto no canto superior esquerdo e
Um segundo canto no canto inferior direito.

Esta é uma imagem de uma caixa, com todas as partes acima identificadas. Note que estou usando os apelidos dos campos aqui, como você provavelmente usará em seus programas.

O compilador consegue fazer caixas a partir de especificações de largura e altura, ou apenas usando um par de pontos de coordenadas. Tudo o que você tem que fazer é escrever algo assim:

FAÇA uma caixa COM tal largura E tal altura.
FAÇA uma caixa COM essa coordenada E essa outra coordenada.
FAÇA uma caixa USANDO esse lado esquerdo E essa parte de cima E esse lado direito E essa parte de baixo.

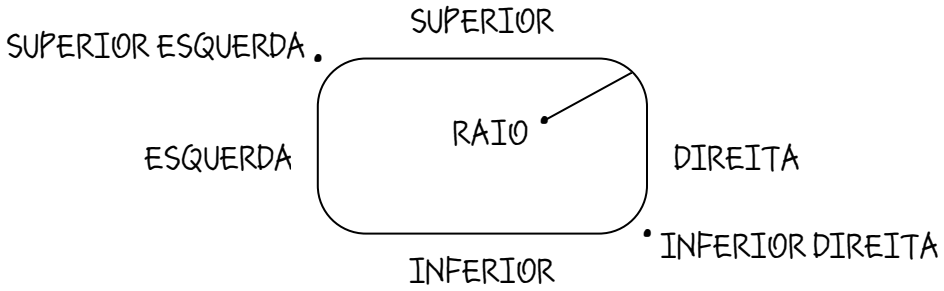
O compilador consegue criar uma caixa. Bem como obter a largura, altura o ponto central entre outras coisas. Ele também consegue dizer se uma caixa ESTÁ DENTRO ou se ela ESTÁ TOCANDO outra caixa. E se um ponto está DENTRO, FORA ou na BORDA da caixa. Sem mencionar todas as outras "Transformações Gráficas" as quais você pode saber mais em outra parte mais abaixo neste glossário.

CAIXAS ARREDONDADAS

Uma "caixa arredondada" é uma caixa com cantos arredondados. O compilador usa essas caixas arredondadas no ambiente de trabalho, nos menus, abas e muitos outros lugares. Aqui está a definição:

Uma caixa arredondada é uma caixa que tem
uma coordenada esquerda, uma coordenada superior, uma coordenada direita, uma coordenada inferior,
um canto superior esquerdo na coordenada esquerda, um canto inferior direito na coordenada direita e
um raio.

Esta é uma imagem de uma caixa arredondada, com todas as partes acima identificadas. Observe que estou usando os apelidos dos campos aqui, da mesma forma que você provavelmente irá usar em seus programas.



O compilador consegue fazer caixas arredondadas a partir de especificações de largura e altura, ou apenas usando um par de pontos de coordenadas. O compilador consegue criar uma caixa arredondada até mesmo usando outra caixa como base. O esquema de funcionamento é mostrado abaixo:

FAÇA uma caixa arredondada com tantas polegadas de largura POR tantas polegadas de altura COM um raio de tantas polegadas.
FAÇA uma caixa arredondada USANDO esta localização E esta outra localização E este raio.
FAÇA uma caixa arredondada USANDO esta coordenada esquerda E esta coordenada superior E esta coordenada direita E esta coordenada inferior E este raio.
FAÇA uma caixa arredondada USANDO esta caixa E este raio.

O compilador consegue DESENHAR uma caixa arredondada. Bem como obter a largura, altura o ponto central entre outras coisas. E se um ponto está DENTRO, FORA ou na BORDA da caixa. Sem mencionar todas as Transformações Gráficas de costume.

CAMPOS

Um record é uma coleção de itens que contém dados intimamente relacionados. Cada item é considerado um "campo". Campos são definidos como parte do record que os contém, e podem ser separada por vírgulas, ponto e vírgula ou pelas palavras E e OU. Consulte o exemplo abaixo:

Uma pessoa é algo com
um nome e
um endereço postal;
um byte chamado gênero ou
um byte chamado sexo no gênero;
32 bytes, e
um cônjuge (referência).

O primeiro campo é definido com apenas um artigo indefinido,UM e um tipo, NOME. Pense neste campo como o nome da pessoa.

O segundo campo inclui um ENDEREÇO POSTAL, que é nada mais do que uma variável do tipo texto. Logo, em algum lugar do seu código você precisará acrescentar a definição:

Um endereço postal é um texto.

O terceiro campo é definido da mesma forma que o primeiro campo, mas com um nome imposto a ele devido a cláusula CHAMADO. Esse campo é o gênero da pessoa. Normalmente, você só usará esssa forma quando o tipo de um campo não tiver nada a ver com o nome do tipo (Ninguém costuma pensar em gênero em termos de bytes).

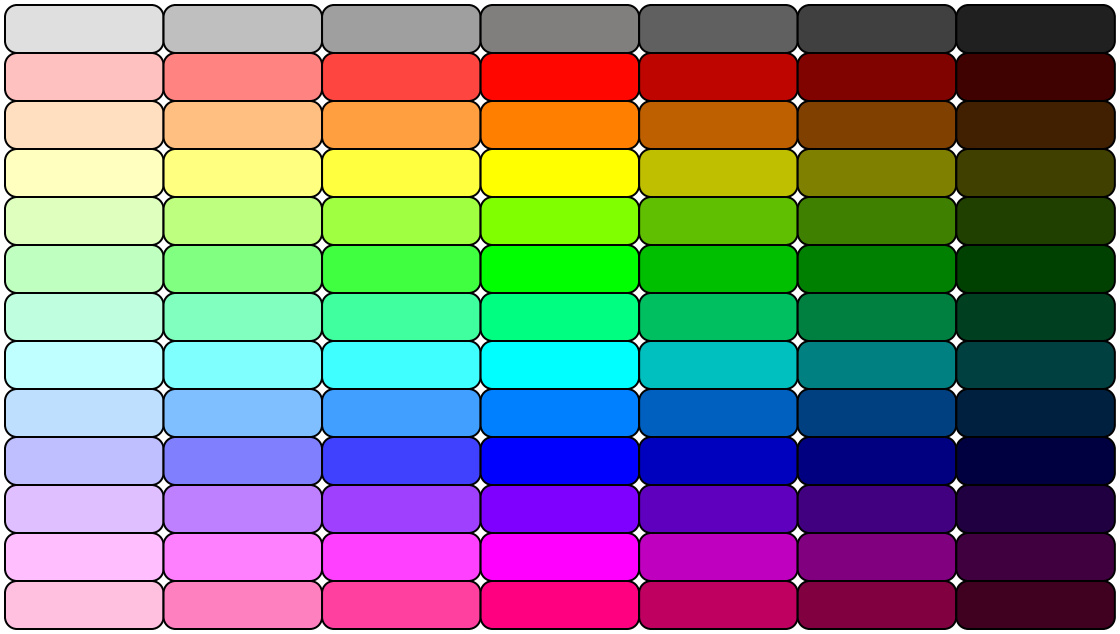
O quarto campo usa a palavra chave NO para redefinir o terceiro campo, dando-lhe um novo nome. Asobreposição de tipos de dados deve ser compatível para que coisas como esta funcionem.

O quinto campo é enchimento. Ele não tem nome e não pode ser acessado diretamente.

O último campo é parecido com o segundo, onde se assume que a variável CÔNJUGE é um tipo definido em outro lugar. A tag (REFERÊNCIA) serve para informar que o termo CÔNJUGE não faz "parte" da pessoa e não deve ser destruído automaticamente quando a pessoa for.

CORES

Uma cor tem uma matiz, uma saturação e um brilho. A paleta padrão de cores inclui clores claras, brancos, pretos, sete cinzas não saturados e oitenta e quatro cores totalmente saturadas com diferentes graus de claridade, como mostrado aqui:



O compilador possui uma variável global para cada uma dessas cores no arquivo o cérebro. "A cor cinza mutio claro", por exemplo, ou "a cor azul escura". Você deve omitir o adjetivo em tons normais, como em "a cor vermelha".

Você também pode inventar suas próprias cores como esta:

FAÇA uma cor USANDO uma tonalidade E uma saturação E um brilho.

Matizes variam de 0 a 3600. O compilador usa múltiplos de 300 na paleta, começando com a cor vermelha no valor 0. Saturação e brilho podem ter qualquer valor entre 0 e 1000.

COMENTÁRIOS

Existem 3 tipos de comentários disponíveis no compilador. Aqui está uma descrição exata de cada um.

Um "comentário" é qualquer coisa entre o uma barra invertida e o final de uma linha::

\ este é um comentário que termina com o byte CR

O editor de texto exibe comentários na cor do azul claro para que sejam fáceis de encontrar.
Caso você deseje alterar a cor dos comentários, será necessário recompilar o programa, mudando a seguinte linha de código:
Utilize a cor azul claro como cor dos comentários.

Os comentários podem começar em qualquer lugar da linha, mas terminam quando a linha termina. No entanto, você pode incluir ou excluir blocos inteiros do código selecionado usando os comandos Comentar e Descomentar no meu editor.

Existe outro tipo de comentário que é chamado de "observação. Ele não possui uma cor diferenciada nem é afetado pelos comandos mencionados acima. Exemplo:

[bytes imprimíveis]

Onde "imprimível" significa qualquer byte da tabela ASCII, exceto caracteres com valores de 0 a 31, o byte de exclusão, e os bytes indefinidos 129, 141, 143, 144, 157.

Observações podem ser colocadas em qualquer lugar, mesmo no meio de uma frase. Mas a fim de evitar erros, as observações também não podem ocupar mais de uma linha. Conforme dito antes, elas não são realçadas. Então use como observação e não como comentário.

O termo "ruído", se refere a todos os caracteres da tabela ASCII cujo valor esteja entre 0 e 31, o byte de espaço, o byte de exclusão, os bytes indefinidos 129, 141, 143, 144, 157, e o byte de espaço rígido. O compilador reconhece estes bytes como separadores, claro, mas, fora isso, não faz nada com eles.

CONDIÇÕES

Uma declaração "condicional" é uma declaração com duas partes. A primeira parte determina as condições sob as quais a segunda parte poderá ser executada. Aqui estão alguns exemplos:

Se a coordenada não estiver dentro da caixa, alerte o usuário.
Se o número for maior que 3, diga "Isso é muito"; saia.
Se o botão esquerdo do mouse estiver sendo pressionado, coloque a coordenada do ponteiro em uma coordenada; repita.
O formato geral é:

Se isso acontecer, faça isso; isto; aquilo.
A palavra SE é necessária. "Isso" representa uma chamada implícita para uma rotina decisora. Se a rotina retornar um "sim", todos os comandos após a vírgula serão executados. Se a rotina retornar um "não", o programa executará a próxima linha.

Observe que os comandos imperativos nos comandos condicionais são separados por ponto e vírgula, e não por pontos, porque o primeiro ponto encontrado indica o fim da instrução.
A nãos er que o ponto esteja num comentário ou dentro de aspas duplas, é claro.

Observe também que se você utilizar palavras negativas na chamada implícita do decidor, essas palavras serão descartadas ou serão modificadas adequadamente. Nesse caso o decisor que será chamado é o que não contém as palavras negativas, a única coisa que acontece é que o valor de retorno dele será invertido. Por exemplo, digamos que você use o seguinte comando:

Se a coordenada não estiver dentro da caixa, [...]

Primeiramente, o compilador vai descartar o não. Logo em seguida, ele vai decidir se a coordenada está na caixa. A resposta obtida será então invertida (Se a resposta for sim, então o retorno será um não e vice-versa). Eu sei que parece complicado, mas na verdade não é. E até onde testamos, tem funcionado muito bem, já que os decisores apenas retornam sim ou não. Consulte o tópico Decisores para mais informações.

Por último, lembre-se: o compilador não aceita o aninhamento de declarações condicionais. A filosofia da linguagem entende que esse tipo de declaração acaba por ser sempre desnecessário (por poder ser expresso de outra forma) e quase sempre incerto (gerando resultados indesejados). Não há nenhuma declaração condicional aninhada em todas as mais de 25 mil linhas de código deste projeto. E se você parar pra pensar que este é o compilador de língua portuguesa mais avançado do mundo, verá que elas realmente não são nenhum pouco necessárias. Na verdade, cada uma das declarações condicionais do compilador cabe numa linha só. Pense nisso.

O CONSOLE

Um "console" é uma interface de texto puro, com a qual você pode interagir de forma limitada. O console padrão parece mais ou menos com a imagem abaixo:

```
Seja bem-vindo ao PAL - Portuguese Compiler And Linker. Qual é o seu nome?
> Dr. Gerry
Bom dia, Dr. Gerry. Estou pronto para a minha primeira lição.
```

O console pode ser ativado a qualquer momento. Ele ocupa toda a tela e usa a fonte padrão na cor preta no fundo cinza muito claro.

Você pode conversar com o usuário no console usando instruções como estas:

```
LEIA algo USANDO O CONSOLE PADRÃO.
MOSTRE algo NO CONSOLE PADRÃO.
```

Você também pode escrever no console sem avançar para a próxima linha:

```
MOSTRE algo NO CONSOLE PADRÃO SEM PULAR LINHAS.
```

Que é útil para "prompts", como o símbolo > no exemplo acima.

O console padrão está sempre disponível, mas ele vai aparecer na tela somente quando você precisar informar alguma coisa para ele ou exibir alguma coisa nele. Uma vez exibido, o console permanecerá visível até que você mande o programa exibir outra coisa e atualizar a tela.

O console lembra tudo o que ele exibe e rola automaticamente para cima quando a parte inferior da tela é alcançada. Você pode usar as teclas Home, End, Page Up, Page Down, e o botão direito do mouse para rolar manualmente.

COISAS
Uma das palavras mais imporantes do compilador é a palavra "coisa". Sempre que o compilador lê a palavra coisa, ele cria um registro especial de tamanho dinâmico, além de criar um registro especial de lista encadeada, assim você pode criar uma lista de coisas. No nosso programa de exemplo, usamos o seguinte comando:

Uma obra de arte é uma coisa que tem uma URL e uma pintura.

Mas a definição pode ser expandida para ficar assim:

Uma obra de arte é um ponteiro para um registro de obra de arte.
Uma registro de obra de arte é um registro que tem uma próxima obra de arte, uma URL e uma pintura.
Algumas obras de arte são algumas coisas que têm uma primeira obra de arte e uma última obra de arte.

Obviamente você não sabia disso. De qualquer forma, isso permite que você utilize os comandos:

ADICIONE uma coisa NO FINAL de algumas coisas.
ADICIONE algumas coisas NO FINAL de outras coisas.
INSIRA uma coisa DEPOIS de outra coisa DENTRO DE algumas coisas.
INSIRA uma coisa ANTES de outra coisa DENTRO DE algumas coisas.
INSIRA algumas coisas DEPOIS de outras coisas DENTRO DE uma coisa.
INSIRA algumas coisas ANTES de outras coisas DENTRO DE uma coisa.
MOVA uma coisa DE algumas coisas PARA outras coisas.
MOVA algumas coisas PARA outras coisas.
ADICIONE uma coisa NO INÍCIO de algumas coisas.
ADICIONE algumas coisas NO INÍCIO de outras coisas.
REMOVA uma coisa DE outras coisas.
INVERTA algumas coisas.

Existe também uma função que permite que você "coloque a quantidade de algumas coisas" em uma contagem. Tudo o que você precisa lembrar é de CRIAR e DESTRUIR cada uma de suas coisas. Consulte "Gerenciamento de memória" para mais informações.

CONSERTANDO FALHAS E ERROS

Um dos princípios que regem a linguagem da Sociedade Osmosiana é: Depuradores são coisa de frangotes.

Se você precisa de uma ferramenta especial para ajudá-lo a consertar seu código, algo está seriamente errado. Ou você está esquecendo de testar seu programa conforme vai acrescentando as funções nele, ou o seu código está irremediavelmente complexo. Ou talvez você esteja na profissão errada.

Dir-vos-ei agora o que fazem os Mestres Osmosianos quando confrontados com um erro na execução do código.

Antes de tudo, eles oram ao bom Deus solicitando sua orientação. Em seguida, eles consideram excluit totalmente a funcionalidade problemática, dessa forma eles se livram de uma vez do problema, e evitam o famoso Feature creep. Em seguida, eles estudam o código, esperando conseguir "discernir" qual é o problema. Se o bug não foi encontrado, eles escolhem um local que seja apropriado para que o programa emita um som de alerta. Se eles não ouvirem o som durante a execução do programa, então tem algo errado na rotina, e eles verificam. Caso eles escutem o som, o comando é movido para próxima rotina, ou para o próximo comando da rotina. O console também pode ajudar muito.

Nos raríssimos casos, em que várias repetições dos procedimento descrito acima falham em providenciar uma conclusão aceitável, os Mestres da Sociedade OSmosiana escolhem outro lugar no código e inserem o seguinte comando:

ANALISE isso.

Onde "isso" pode ser uma caixa, byte, cor, bandeira, fonte, linha, número, par de coordenadas, ponteiro, proporção, local, texto ou wyrd. Quando eles executam o código modificado, a caixa de mensagem do Windows aparece contendo uma pista. A aparência horrenda da caixa os motiva a rezar mais e renovar a determinação de resolver o problema, E assim armados, eles voltam ao primeiro passo.

Ofereço a minha própria existência como prova da suficiência destas técnicas. E estou confiante que todos os futuros erros — exceto talvez, um inesperado Ciclo infinito de Mobius - poderão ser gerenciados da mesma forma.

DECISORES

Um decisor nada mais é do que uma rotina que retorna "sim" ou "não" para determinados comandos. Exemplo:

Para decidir se uma coordenada está dentro de uma caixa:
Para decidir se um número é maior que outro número:
Para decidir se algo está selecionado em um texto:
As rotinas decisoras sempre começam com as mesmas três palavras. O formato é:

PARA DECIDIR SE alguma coisa:
Esse "alguma coisa" deve seguir as regras existentes para nomes de rotina e tipicamente incluirá um verbo como o É, ESTÁ, PODE, FAZ, DEVE, ETC. Perceba que a conjugação dos verbos não importa muito.

O ideal é sempre criar decisores "positivos". Ou seja, se você quiser evitar dores de cabeça no futuro usar as palavras NÃO, NUNCA, NENHUM, NADA e similares. Não se preocupe, caso seja realmente necessário utilizar tais temos, você pode. Mas na prática o compilador vai meio que desfazer tudo que você fez.

Por exemplo, se você criar uma rotina que "decide se uma coordenada está dentro de uma caixa", o compilador automaticamente vai saber dizer se "uma coordenada NÃO está dentro de uma caixa. Isso funciona pra quelauqer tipo de rotina decisora. Para saber mais, procure os exemplos disponíveis nos arquivos de código fonte.

Os decisores funcionam basicamente da mesma forma que as rotinas condicionais e imperativas. No entanto você não pode usar o comando SAIA em um decisor, por motivos óbvios, bem como qualquer outra forma que faça você sair sem dar alguma resposta. Um decisor sempre deve terminar com as palavras "DIGA SIM" ou "DIGA NÃO".

LISTA DE DECISORES DISPONÍVEIS NO SISTEMA

Decisores são uma boa forma de deixar o seu programa (e o compilador) mais inteligente. Você pode (e deve) integrá-los ao sistema. Existem 138 decisores no sistema. Neste exato momento, com certeza devem haver bem mais. Eis uma amostra dos principais decisores:

É
É ALFANUMÉRICO
É ALGUMA CONSOANTE
É ALGUM DÍGITO
É ALGUMA LETRA
É ALGUM SÍMBOLO
É ALGUMA TECLA NUMÉRICA
É ALGUMA TECLA ALFABÉTICA
É ALGUMA TECLA DE COMANDO
É ALGUMA TECLA DO TECLADO NUMÉRICO
É ALGUMA TECLA DE SÍMBOLOS
É ALGUMA VOGAL
ESTÁ ENTRE
ESTÁ EM BRANCO
É INEXISTENTE
ESTÁ FECHADO
ESTÁ SENDO PRESSIONADO
ESTÁ VAZIA
É PAR
É DIVISÍVEL INTEIRAMENTE POR
É MÚLTIPLO DE
COMEÇA COM
É MAIOR [DO] QUE
É MAIOR [DO] [QUE] OU IGUAL A
ESTÁ DENTRO
É MENOR DO QUE
É MENOR [DO QUE] OU IGUAL A
É MUITO CLARO\ESCURO
É COMO
É UM NÚMERO NEGATIVO
É UM NUMERO POSITIVO
É RUÍDO
É ÍMPAR
ESTÁ LIGADO
É IMPRIMÍVEL
ESTÁ MARCADO COMO SOMENTE LEITURA
ESTÁ DEFINIDO [FLAGS]
É SIMBÓLICO
ESTÁ TOCANDO
ESTÁ ACIMA [DE]
É MUITO ESCURO
É MUITO CLARO
É UM ESPAÇO EM BRANCO
ESTÁ ENTRE
TERMINA COM

Alguns desses funcionam com apenas um tipo de dado, é claro, mas outros trabalham com muitos. E se você leu o tópico Decisor, você sabe que o compilador também aceita as variantes negativas desses mesmos decisores. Por favor, não tente memorizá-los. Essa não é a ideia. Apenas diga o que você quer dizer em seu programa, e se o compilador não entender, adicione o decisor à coleção de rotinas e me deixe o compilador mais inteligente.

DESENHANDO
Você pode dizer ao compilador coisas como:

DESENHE uma coisa.
DESENHE uma coisa USANDO uma cor.
DESENHE uma coisa USANDO uma cor na borda E uma cor no preenchimento.
DESENHE uma coisa DENTRO DE uma caixa USANDO uma fonte E uma cor.
DESENHE uma coisa NO CENTRO DE uma caixa USANDO uma fonte E uma cor.

E o compilador irá renderizar tudo numa "tela virtual", ou seja, uma tela invisível do mesmo tamanho e formato que a tela do monitor. Dessa forma, quando você utilizar o comando:

ATUALIZE A TELA.

O compilador irá substituir o conteudo da tela pelo conteúdo da tela virtual em um piscar de olhos. Na verdade, mais rápido que isso. Quer dizer, às vezes não. Se você usar o comando:

ATUALIZA A TELA USANDO [o conteúdo de] uma caixa.

O compliador vai transferir apenas os pixels que estiverem dentro da caixa.

As regras para impressão funcionam de forma diferente. Nesse caso, o compilador usa "a tela da impressora" e envia os desenhos para um dispositivo de impressão assim que você finaliza página. Consulte a seção "Imprimindo" para mais detalhes.

Para ajustar suas coordenadas, você pode:

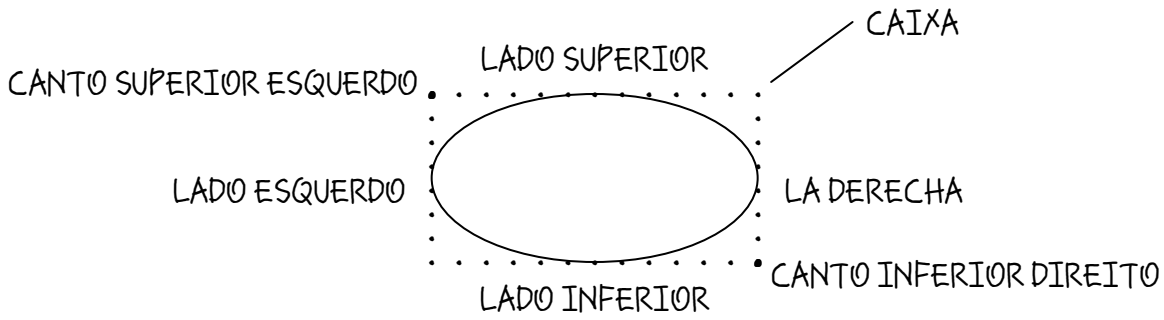
DEFINIR uma coordenada COMO ORIGEM DO DESENHO.

Você também pode evitar o desenho em certas áreas da tela usando o recurso de "fita crepe". Consulte o tópico "Fita Crepe" para descobrir como fazer isso.

ELIPSES
Infelizmente o Windows não permite círculos nem elipses — ele só desenha retângulos redondos dentro de caixas sem bordas. O que explica essa definição incomum de "elipse" que está presente no cérebro:

Uma elipse tem uma caixa.

Esta é uma imagem de uma elipse, com suas partes identificadas. Note que você pode obter os campos individuais da caixa da elipse usando o meu recurso de acesso aos "campos profundos", recurso este que é descrito sob o tema "Possessivos" neste glossário.



O compilador pode fazer elipses de várias maneiras diferentes. Usando especificações de largura e altura. Ou usando um par de coordenadas. Ou todas as quatro coordenadas. Tudo o que você tem que fazer é escrever algo assim:

FAÇA uma elipse USANDO tal valor na largura E tal valor para a altura.
FAÇA uma elipse USANDO essa coordenada E essa outra coordenada.
FAÇA uma caixa USANDO esse lado esquerdo E essa parte de cima E esse lado direito E essa parte de baixo.

Assim como caixas, o compilador consegue DESENHAR uma elipse. Bem como obter a largura, altura o ponto central entre outras coisas. Inclusive detectar se uma coordenada ESTÁ DENTRO (ou fora) de uma elipse ou se a coordenada ESTÁ na borda de uma elipse. Sem mencionar todas as outras "Transformações Gráficas" as quais você pode saber mais em outra parte mais abaixo neste glossário.

FLUXO PADRÃO DE ENTRADA E SAÍDA

Você pode trabalhar diretamente com o mouse usando instruções como estas:

ARMAZENE a posição do mouse EM uma localização.
SE o botão esquerdo do mouse FOR PRESSIONADO, ...
SE o botão direito do mouse FOR LIBERADO, ...

Mas você provavelmente não vai precisar, a menos que esteja monitorando o mouse, enquanto o usuário arrasta alguma coisa pela tela. Na maioria das vezes, você simplesmente irá responder aos vários eventos do tipo "clique" que são enviados para o seu gerenciador de eventos.

Você pode trabalhar diretamente com o teclado usando instruções como:

SE a tecla Esc FOR PRESSIONADA, ...
SE a tecla Shift FOR LIBERADA, ...

Mas novamente, você provavelmente não vai precisar usar esses comandos, porque o Windows funciona melhor se você apenas responder aos eventos de "tecla pressionada" que forem enviados para seu gerenciador de eventos. Você pode encontrar todas as teclas globais procurando por "é uma tecla igual a".

Você pode trabalhar diretamente com a tela usando a variável quadro da tela e essa é uma variável global única:

A tela tem uma caixa, uma altura de pixel e uma largura de pixel.

Mas você não deveria. Em vez disso, é melhor que você use a tela virtual, para aí sim passar o conteúdo da tela virtual para a tela real usando o comando:

ATUALIZE A TELA.

Consulte o tópico Desenhando para obter mais informações. Mas não hesite em usar a caixa da tela e todos os campos da tela ao inicializar suas coisas.

PERCORREDORES

Um "percorredor" é um registro usado para analisar texto. Para entendê-lo, você deve estar confortável com o modo de funcionamento do tipo texto (strings) e subtextos (substrings). Se você não estiver familiarizado com estes termos, procure-os neste glossário e analise as manipulações de texto que fizemos no nosso programa de exemplo.

Eis a definição de "percorredor":

Um percorredor possui
um subtexto original,
uma subtexto fonte e
uma caractere de subtexto.

Eis o que acontece quando o comando abaixo é invocado:

COLOQUE um percorredor em um texto.

O compilador define o subtexto original e o subtexto fonte de forma que eles possam abranger todo o texto. Então ele posiciona o caractere de subtexto no subtexto fonte — o que faz com que o subtexto fonte comece com valorem branco, mas pronto para receber um novo valor. Depois de colocar um percorredor no texto, você pode usar o seguinte comando:

AVANCE um percorredor.

O compilador adiciona um no primeiro byte do subtexto fonte e mais um no último byte do subtexto fonte. Isso faz com que o subtexto fonte fique mais curto enquanto que o caractere de subtexto fica mais longo, permitindo que você processe o texto um byte de cada vez. Quando você quiser limpar o caractere de subtexto antigo e começar um novo, você só precisa utilizar o comando:

POSICIONE o caractere de subtexto do percorredor no subtexto fonte do percorredor.

Você também pode escrever suas próprias rotinas para MOVER um percorredor mais de um byte por vez, assim como acontece nas rotinas de "verificação ortográfica" e de "quebra de linha", sem mencionar as rotinas de análise de código-fonte. Pesquise para mover um percorredor para encontrar os exemplos.

EVENTOS/COMANDOS RECEBIDOS PELO SISTEMA

O Windows insiste que usemos seu modelo complicado e não processual juntamente com as suas centenas de mensagens e códigos absurdos. Felizmente, o compilador inclui definições que reduzem essa monstruosidade a apenas dez simples eventos que podem ser tratados de uma forma puramente processual. Eis o código:

Um evento é uma coisa com
uma categoria,
um detector de shift, uma detector de ctrl, uma detector de alt,
uma coordenada,
uma tecla e um byte.

Categoria nada mais é do que uma das seguintes palavras:

ATUALIZAÇÃO DE TELA — Esse tipo de evento informa que é hora de redesenhar a tela. Culpe o Windows por isso.
MOVIMENTO DE CURSOR — O cursor se moveu. Esse tipo de evento é muito usado para redimensionar formas e objetos.
PRESSIONAMENTO DE TECLA — O usuário digitou algo. Acho que nem preciso explicar nada.
CLIQUE ESQUERDO (ou apenas CLIQUE) — O botão esquerdo do mouse acabou de ser presssionado. Muito usado em botões e componentes similares.
CLIQUE DUPLO — O usuário clicou duas vezes. Se você usa o Windows então já sabe que geralmente esse comando é utilizado para abrir ou executar arquivos.
CLIQUE DIREITO — O botão direito do mouse acabou de ser pressionado. No compilador, esse é o comando que inicia a rolagem da página.
CLIQUE DIREITO DUPLO — O usuário deve ter clicado por engano. Que tal colocar darmos a ele ovo de páscoa para celebrar?
DESATIVAÇÃO — O usuário mudou para outra janela. O compilador lida com isso internamente.
ATIVAÇÃO — O usuário retornou para o seu programa. O compilador lida com isso internamente.
ENCERRAMENTO — Informa que o programa está sendo encerrado. Inserido internamente. Você não deveria ver esse tipo de evento.

Os detectores de ctrl e alt e shift indicam o estado das teclas correspondentes no momento do evento (o detector é acionado se a tecla for pressionada).

A coordenada é a posição do mouse no momento do evento.

A tecla e seu byte ASCII equivalente (se houver) aplicam-se apenas à Teclas Pressionadas.

Esta é a estrutura de um programa orientado a eventos:

Para que o programa seja executado:
Inicie o programa.
Gerencie os comandos do usuário.
Feche o programa.
Para gerenciar os comandos do usuário:
Remova o evento da fila.
Se o evento não existir, ignore-o.
Gerencie o comando.
Repita.

Para gerenciar um comando:
Se a categoria do evento é "atualização de tela", [encaminhe a atualização] saia.
Se a categoria do evento é "pressionamento de tecla", [encaminhe a atualização] saia.
Se a categoria do evento é "clique do mouse", [encaminhe a atualização] saia.
[...]

seu programa

Se não houver eventos sendo executados, a rotina de "desenfileirar" continuará verificando a situação junto ao Windows até que o seu usuário distraído faça alguma coisa. Para encerrar o programa, você deve utilizar um dos seguintes comandos:

RENUNCIAR AO CONTROLE.
DELEGAR O CONTROLE.
ENTREGAR O CONTROLE.
ABANDONAR O CONTROLE.
LARGAR O CONTROLE.

Em algum lugar do seu código. Geralmente em um dos seus gerenciadores de eventos. Esta rotina configura as coisas para que o próximo evento que você "desenfileirar" seja nulo, terminando o seu loop de gerenciamento de eventos.

EXPRESSÕES

Uma "expressão" é como uma oração subordinada em uma frase complexa. É uma frase que deve ser reduzida, separadamente, antes que a instrução que a contém possa ser totalmente compreendida. Se você, por exemplo, dizer:

Coloque a altura menos 1 vez a contagem em um número.
O compilador converte a frase a altura menos 1 vez a contagem para algo muito mais simples antes de pensar em colocar qualquer coisa em qualquer lugar.
O compilador considera como expressão qualquer frase com uma ou mais das seguintes palavras:

MAIS, MENOS, VEZES, DIVIDIDO POR, ou EM SEGUIDA.

Os quatro primeiros são operadores aritméticos padrão, mas você pode aplicá-los a outras coisas também. O último é usado principalmente para unir palavras e textos.
Deixe-me explicar como o compilador simplifica expressões com alguns exemplos.

Digamos que o compilador encontre a palavra MAIS entre a palavra couve e a palavra flor. Ele busca por uma rotina que diga como adicionar uma couve a uma flor, e então usa essa rotina para reduzir a expressão. Se ele encontrar uma couve MENOS uma flor, ele procura uma rotina para subtrair uma flor de uma couve. Para calcular um couve VEZES uma flor, ele usa a rotina para multiplicar uma couve por uma flor. E para poder calcular uma couve DIVIDIDO POR uma flor, ele busca e usa a rotina para dividir um couve por uma flor.

O compilador lida com o último operador um pouco diferente, já que o objetivo neste caso é sempre para anexar uma texto no final de outro texto. Então, por exemplo, se o compilador encontrar a expressão EM SEGUIDA entre, digamos, uma palavra e um número, o compilador busca por uma rotina para converter um número em um texto, aplica a rotina no número e então adiciona o número convertido ao final do texto.

É claro que é possível extender essa capacidade. Mas use com moderação.

FUNÇÕES

Uma "função" é uma rotina que extrai, calcula ou deriva algo usando uma variável. Alguns exemplos são:

Para colocar a linha inferior de uma caixa em uma linha:
Para colocar a altura de um polígono em uma altura:
Para colocar a posição do mouse em uma posição:

Há dois formatos muito semelhantes para funções. O primeiro é:

PARA COLOCAR UM nome DO tipo EM um nome DE tipo:

E o segundo é:

PARA COLOCAR UM nome DO nome EM um nome DE tipo:

Ambas as formas são facilmente reconhecidas porque elas incluem as palavras COLOCAR e EM com um possessivo entre elas. O primeiro formato é o mais comum e é usado com tipos e variáveis normais. O segundo é usado com variáveis únicas globais e pseudo-variáveis.

O que é especial sobre funções é que você pode usar suas partes possessivas como se elas fizerem referência a campos reais em um registro. Por exemplo, dadas as funções acima, você pode obter a linha inferior da caixa como se ela já tivesse sido definida no tipo de registro da caixa. Você pode solicitar a altura do polígono e veremos que ela é calculada quando você precisar. E você pode dizer a localização do mouse e o compilador busca a lozalização para você, mesmo que a variável mouse não tenha nenhum item localização dentro dela.

Nem preciso dizer o quão útil é essa função. Mas tente não abusar dela. Seja sábio.

Consulte o tópico Possessivos para mais informações.

GERENCIAMENTO DE MEMÓRIA

O compilador gerencia toda a memória necessária para os tipos de dados estáticos — como bytes, wyrds, números, ponteiros, bandeiras e a maioria dos registros. O compilador também gerencia as strings (texto), uma vez que elas são usadas com frequência e seu comportamento é previsível.

Mas quando você define um tipo dinâmico de dados, como uma "coisa", você se torna o único responsável por qualquer memória usada pelo tipo.

Normalmente, você vai programar uma rotina CRIAR para inicializar cada tipo dinâmico que você definir. Nessa rotina, você atribuirá memória para a coisa. O esquema de funcionamento é mostrado abaixo:

```
ALOQUE MEMÓRIA PARA algo.
ALOJE MEMÓRIA PARA algo.
ATRIBUA MEMÓRIA PARA algo.
RESERVE MEMÓRIA PARA algo.
SEPARE MEMÓRIA PARA algo.
```

Você também pode codificar uma rotina DESTRUIR para cada tipo, com uma linha como:

```
DESALOQUE A MEMÓRIA DE algo.
DESALOJE A MEMÓRIA DE algo.
REMOVA A MEMÓRIA DE algo.
LIMPE A MEMÓRIA DE algo.
ESVAZIE A MEMÓRIA DE algo.
```

Mas se você não quiser, pode usar a rotina padrão do sistema. Apesar da rotina não ser ececutada automaticamente, ela existe, basta usar o comando certo. As rotinas DESTRUA que podem ser chamadas desta maneira:

```
DESTRUA algo.
```

Observe que a rotina DESTRUA não só destrói a coisa em si, mas também quaisquer outras coisas que estejam dentro da coisa, como campos e listas. A menos que, que você defina esses campos como "(REFERÊNCIA)".

HABILIDADES BÁSICAS

Eu não acho que eu esteja me gabando quando digo que o compilador é rápido e preciso. Sem falar que a linguagem possui um âmbito de aplicação bem amplo. Por exemplo, para zerar o valor interno de alguma variável, basta usar o comando:

LIMPE a variável.

O compilador também é capaz de entender instruções do tipo:

COLOQUE isso DENTRO disso.

Ou ainda:

TROQUE isto POR isso.

O compilador é capaz de criar cópias de variáveis dinamicamente alocadas, como figuras e polígonos, usando comandos como:

COPIE isto PRA DENTRO disso.

E também:

CONVERTA uma coisa EM outra coisa.

Às vezes até implicitamente. Digamos, por exemplo, que você queriaadicionar uma fração no final de uma sequência de caracteres, usando uma expressão como esta:

uma sequência de caracteres seguida de uma proporção

O compilador é esperto o bastante para usar automaticamente a rotina de CONVERTER uma proporção EM um texto antes de efetuar a junção usando a rotina de ADICIONAR um texto NO FINAL DE outro texto. Fantástico. SIm, nós pensamos em tudo (ou quase!).

IMAGENS

Uma "imagem", para o compilador, é uma imagem feita de pixels. Geralmente, milhões de pixels. Imagens são um tipo de dado muito difícil de se trabalhar. Existem dezenas de formatos e padrões. Felizmente, o compilador já tem rotinas para lidar com os formatos mais comuns como png e jpg. Eis a definição:

Uma imagem é uma coisa [com coisas que você não quer saber].

Aqui estão três imagens de amostra:



Peço desculpa pela qualidade destas imagens. Eu as descobri nos antigos arquivos quando eu estava pesquisando sobre a fundação da Sociedade Osmosiana.

Você pode criar uma imagem de várias maneiras. Você pode carregar um a partir de um endereço que contenha uma imagem BMP, JPG, GIF ou alguma outra imagem no formato padrão. Você pode colocar uma imagem em um depósito (buffer) e usar esse buffer como a fonte para sua imagem. Ou você pode pegar uma foto na internet através de uma URL. Você também pode criar uma imagem desenhando alguma coisa e, em seguida, "extraindo" a parte que você quer. Este é o formato geral:

CRIE uma imagem DE algo.

Assim que você tiver uma imagem, você pode DESENHÁ-la Ou aplicar as Transformações Gráficas nela. Ou usá-la como um modelo para uma verdadeira obra de arte, como fizemos com o programa de exemplo como pintar.

IMPERATIVOS

Um comando "imperativo" é um comando não condicional dentro do corpo de uma rotina. Aqui estão alguns imperativos de amostra retirados do compilador:

Alerte o usuário.
Subtraia 1 da quantia.
Remova o último pedaço do texto.
Coloque a altura da fonte do texto multiplicada por 2 no x da grade.

Os imperativos normalmente começam com um verbo e terminam com um ponto. Mas no meio, entra quase qualquer coisa. Literais. Termos. Expressões. Frases preposicionais. Tudo junto e misturado.

Para criar um imperativo, basta digitar o que você está pensando. Se houver uma rotina que consiga lidar efetuar o comando, o compilador irá executar a mesma. Caso contrário, o compilador exibirá uma mensagem de erro, exigindo que você mesmo crie a rotina.

Eis os onze imperativos básicos presentes no compilador:

DIGA. Este imperativo é utilizado dentro de rotinas decisoras. Para mais informações, consulte o tópico Decisores.

PERCORRA, REPITA, PARE e SAIA. Esses comandos s]ao usado em laços de repetição. Consulte a seção Laços.

CHAMAR e APONTAR. Usado para chamar bibliotecas e funções do Windows. Consulte a seçãoWindows.

EMPLOY, PUSH e INTEL. Comandos que espero que você nunca precise utilizar. Consulte Imperativos Especiais.

PRIVATIZE. Usado somente com "Parâmetros".

Já que os imperativos são apenas palavras que chamam rotinas, você também deve conferir as páginas Rotinas, Procedimentos, Decisores, Funções e Nomes.

IMPERATIVOS ESPECIAIS

Os três "imperativos especiais" do compilador provavelmente estão mais para "imperativos usados para propósitos especiais". Talvez você chegue a usar um deles. Esperamos, em última análise, eliminá-lo completamente. Os outros dois são para nerds.

Um imperativo de "emprego" deve ser a única declaração em uma rotina. Esse tipo de imperativo faz com que o compilador utilize outra rotina no lugar da que foi chamada. Só funciona quando os parâmetros de ambas as rotinas estão na mesma ordem e são do mesmo tipo. Para encontrar exemplos, basta progurar por ": employ". Você pode dar mais de um nome para uma rotina, usando ponto e vírgula no lugar de dois pontos, assim:

```
PARA limpar a tela;
PARA apagar a tela;
Para deixar a tela em branco:
[código aqui]
```

Um imperativo "push" avalia uma expressão e coloca o resultado — que deve ser um valor contendo um, dois ou quatro bytes — na pilha do sistema. Você provavelmente não vai precisar usar ele. Nem eu uso isso. Isso é um resquício dos dias em que o comando CHAMAR ainda estava em desenvolvimento (Consulte o tópico Windows para maiores informações sobre esse comando). Só pra você saber, o formato geral do comando é:

```
PUSH uma expressão.
```

Um imperativo "Intel" insere código de máquina no seu arquivo executável. É possível encontrar exemplos complexos em vários lugares do compilador. É uma pena que o Intel não é uma máquina de pilha. O formato é trivial:

```
INTEL nibble literal.
```

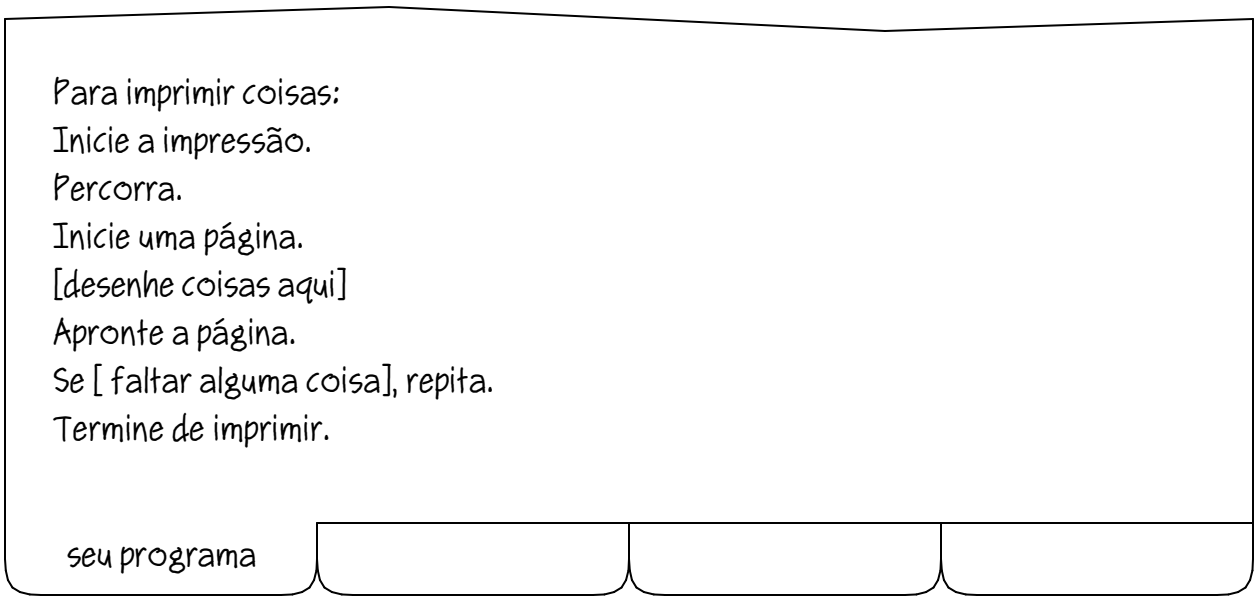
Se você está se perguntando por que razão o compilador não possui um assembler (montador) integrado, a resposta é simplesmente porque não é necessário. Na verdade, há muito pouca linguagem da máquina no meu compilador, e conforme novas funções vão sendo adicionadas, mais e mais desses comandos vão sendo substituídos por português simples. Além disso, um dos princípios dos Mestres da Sociedade Osmosiana é que o melhor assembler sempre foi e sempre será a sua cabeça. É uma forma de manter seu cérebro em dia.

IMPRIMINDO

A rotina de impressão precisa ser aprimorada. Eis alguns detalhes da implementação:

(1) O compilador sempre envia o trabalho de impressão para a impressora padrão do sistema;
(2) O compilador faz uso das rotinas habituais de desenho.
(3) O compilador verifica se o texto que aparece na página de impressão é igual ao conteúdo que está na tela.

Aqui está uma rotina típica de impressão:



"O comando Inicie uma página define o quadro (área de desenho) atual como o quadro da impressora. O comando Apronte a página coloca o conteúdo da página de volta ao quadro da tela virtual. Então posicione quaisquer mensagens de status que você desejar exibir antes ou após essas chamadas.

Você pode usar o comando inicie uma página vertical para ser mais explícito, e você pode iniciar uma página horizontal para trabalhar em modo paisagem. As várias páginas são na verdade caixas, inicializadas pela rotina inicie, que você pode usar para posicionar suas coisas.

Basicamente é só isso.

INTERNET

Essas são as variáveis que você precisará usar para obter arquivos da internet:

Uma URL é um texto.

Um texto de requisição é um texto.

Uma URL é um Localizador Universal de Recursos, por exemplo http://www.osmosian.com, que como você pode ver é apenas uma seqüência de caracteres que segue uma convenção complicada para nomes. Essa convenção foi baseada na tecnologia de análise disponível na época, e que fazia muito sucesso há uns 50 anos.

Umtexto de requisição ou "query string" é um texto com alguns dos seus bytes convertidos em codificações absurdas consistentes com os padrões da internet. Um espaço, por exemplo, se torna uma %20, e uma vírgula se torna %2C.

Você pode converter um texto comum em um texto de requisição usando o comando a seguir:

CONVERTA um texto EM UM texto de requisição.

Você pode acessar um recurso da internet usando o seguinte comando:

COLOQUE uma URL DENTRO DE um texto.

Aqui está um código do nosso programa de exemplo para te lembrar como funcionaque você lembre-se de como funciona

Coloque "http://images.google.com/images?q=" em uma URL.
`Converta o texto do campo de texto em um texto de pesquisa.
Coloque o texto de pesquisa no fim da URL.
COLOQUE a URL DENTRO DE um texto.

Lembra? Não tem nem como esquecer. O compilador analisou o texto, pintou o quadro, atualizou a tela, gerando uma verdadeira Obra de arte.

O WINDOWS

Se você por acaso precisar utilizar bibliotecas e funções internas do Windows, você pode utilizar comandos semelhantes a este:

CHAME "NomeDaDll.dll" "NomeDafunção" COM parâmetro E RETORNE algo.

As cláusulas COM e RETORNANDO são opcionais. Você deve diferenciar maiúsculas e minúsculas no nome da função, usando o mesmo nome que o nome da dll do Windows ou da sua pasta. Textos devem ser passadas por endereço e, em muitos casos, devem terminar com o byte NUL. Use "o primeiro byte do texto" para o endereço, e a rotina a seguir para adicionar um byte nulo no final do texto:

ADICIONE O BYTE NULO no final de um texto.

Em outros casos, o Windows nos fornece não nome de uma função, mas o endereço dela. Você pode chamar essas funções usando uma sintaxe similar:

CHAME um endereço COM esse parâmetro E RETORNE algo.

Às vezes, o Windows precisa que nós forneçamos o endereço de uma de nossas rotinas para que ele possa interromper nosso previsível fluxo processual em algum momento. Você pode usar esta sintaxe para obter o endereço de uma rotina:

APONTE um ponteiro PARA ROTINA nome de rotina.

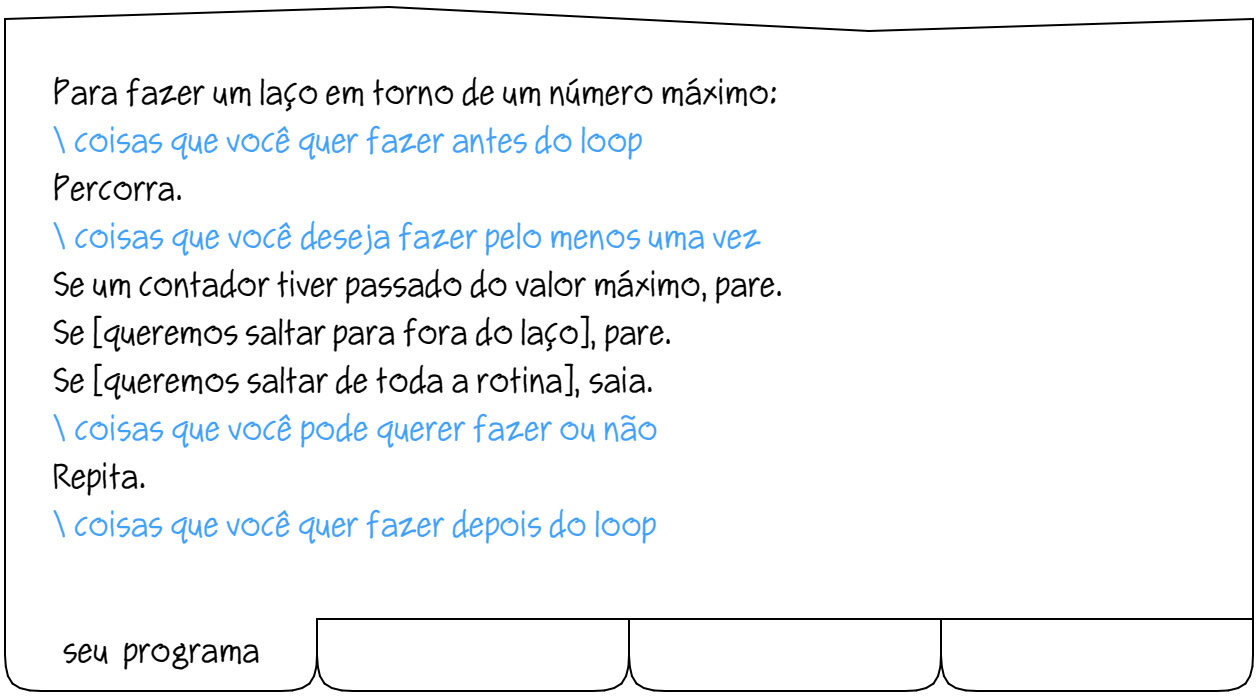
Mas se você for passar o endereço para o Windows, certifique-se de que o cabeçalho da rotina inclua a palavra-chave COMPATIVELMENTE logo após PARA, assim:

PARA COMPATIVELMENTE ...

Se você está trabalhando neste nível ridiculamente baixo, você vai querer verificar as seções "Bits", "Imperativos Especiais" e meu o código fonte para mais informações e exemplos.

LAÇOS

Os comandos específicos para laços são PERCORRA, REPITA, INTERROMPA e SAIA.



O comando PERCORRA não faz nada a não ser indicar o começo do laço. O comando REPITA volta para o comando PERCORRA, se houver um. Se não houver, o comando volta para o topo da rotina. O comando INTERROMPA sai do LAÇO e prossegue para o comando logo após o último comando REPITA. Se não houver nenhum comando REPITA, o comando vai fazer a mesma coisa que o comando SAIA. Você só pode usar um comando PERCORRA em cada rotina, mas não há limites para a quantidade de REPITA, INTERROMPA ou SAIA.

O comando que começa com "Se um contador tiver passado do máximo" chama um decisor especial que avança o conrador e verifica o mesmo logo após. Como o contador é uma nova variável local, quando a rotina é iniciada ele começa valendo zero.

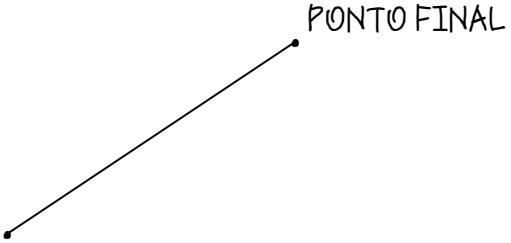
Observe que você pode utilizar os comandos PERCORRA, REPITA e INTERROMPA em um decisor, mas você não pode usar o comando SAIA porque isso interrompe o fluxo de decisão da rotina. Por isso você precisa tomar cuidado para não sair sem querer de um decisor, também. Para sair de um decisor, utilize os comandos DIGA SIM ou DIGA NÃO.

LINHAS

Uma "linha" é um objeto gráfico que começa em um lugar e termina em outro lugar. Mas você já sabia disso. Eis a definição de linha:

Uma linha tem um ponto inicial e um ponto final.

E aqui está uma imagem de uma linha, com suas partes identificadas:



O compilador consegue fazer linhas a partir de dois pontos ou usando quatro coordenadas separadas (2 x e 2 y):

FAÇA uma linha USANDO essa coordenada E essa outra coordenada.
FAÇA uma linha COM este x E este Y E aquele x E aquele Y.

O compilador também tem quatro funções que colocarão uma LINHA ESQUERDA DA caixa ou uma LINHA SUPERIOR de uma caixa ou uma LINHA DO LADO DIREITO de uma caixa ou uma LINHA INFERIOR da caixa em uma linha.

O compilador consegue CRIAR uma linha. Ou encontrar o CENTRO de uma linha. O compilador é capaz de até mesmo de:

DIVIDIR uma linha EM uma linha E outra linha.

Bem no meio. Ele consegue detectar se um ponto está em cima de uma linha. Além de conseguir realizar todas as Transformações Gráficas habituais em linhas, também.

Consulte o tópico Spots para mais informações sobre esses pontos finais.

LISTAGEM

Se você é um fã de compiladores, você vai gostar disso. Se você não for, vire a página.

Você pode produzir uma lista criptografada de todas as definições do compilador utilizando o comando Listar. A listagem é salva como texto na pasta de origem recebe o nome dessa pasta junto com a extensão .lst. A interpretação deste arquivo é deixada para você como um exercício.

Mas vou dar-lhe algumas pistas. E algum incentivo.

A listagem consiste em doze seções distintas com os seguintes títulos: tipos, globais, literais, rotinas, tipo índice, índice global, índice literal, índice de rotina índice, índice de utilidade, importações, arquivos fonte e temporizadores. Cada título é seguido de dois pontos para que você possa pular para qualquer seção usando o comando Encontrar. Aqui está uma pequena amostra da seção rotina:

```
/routine/create [picture]/yes/no/no/no//4/0/00470A48/
/variable/parameter/yes/picture/picture/picture/picture/00000008/no/1/no////
/fragment/prolog/////00000000/00470A48/558BEC/
/fragment/loop/////00000000/00470A4B//
/fragment/push address/picture////00000000/00470A4B/8B950800000052/
/fragment/call internal///allocate memory for [picture]//00000000/00470A52/E8BDA70400/
/fragment/finalize/////00000000/00470A57//
/fragment/epilog/////00000000/00470A57/8BE55DC204000000/
```

Eu recomendo que você estude a rotina listar no compilador antes. Faça um pequeno programa, use o comando listar, e consulte o resultado. Adicione uma linha ou duas, e repita.

Passemos agora ao incentivo.

Se você encontrar algum erro no compilador, mande um email para os criadores e eles lhe enviarão algo bacana como uma camiseta em branco. Se você consegue descobrir como tornar o compilador mais simples sem deixar mais lento, eles vão te enviar uma camiseta personalizada. E se você puder criar uma maneira de fazer o compilador menor, mais rápido e mais poderosa tudo de uma só vez, tenho certeza que eles te enviarão uma camiseta sem manga bordada.

LITERAIS

Um valor "literal" é um valor constante em um programa. O compilador entende sete tipos diferentes de literais, cada um com um formato específico.

Um "número" literal são dígitos, com um sinal opcional, mas sem espaços ou observações:

Exemplos: 0,-2147483648, +2147483647

Uma "Proporção literal" é um número, uma barra e um número sem sinal:

Exemplos: 335/113, 25946/9545, -19601/13860

Um "literal misto" é um literal numérico, um traço e uma proporção sem sinal.

Exemplos: 1-1/2, -2-2/3, 3-3/4

Um texto literal é uma série de caracteres entre aspas duplas. Se você precisar de uma aspa dupla dentro de uma string, coloque duas ao redor dela e pronto. Assim:

"Este é um texto literal com ""aspas duplas em torno disto"" mas não disto"

O único "ponteiro literal" é a palavra-chave NIL. Ele indica um ponteiro vazio ou inválido. Para fazer com que um ponteiro fique vazio (NIL) é só usar o comando ESVAZIAR.

Uma bandeira literal é uma das palavras-chave SIM ou NÃO. Você pode definir uma bandeira para colocar SIM nela, e você pode LIMPAR uma bandeira para colocar NÃO nela.

Um nibble literal é um $ seguido de dígitos hexadecimais. Provavelmente você não precisará utilizá-los. Aqui está uma amostra, de qualquer forma:

$DEDOFEDE

FITA CREPE

Os pintores da vida real geralmente usam fita crepe, dessa forma eles não pintam onde não querem. Da mesma forma que os artistas, você pode usar as rotinas de "máscara" do compilador para restringir a rotina de desenho do compilador. Os comandos são esses:

PROTEJA O INTERIOR disso. / PROTEJA A PARTE DE DENTRO disso. /
PROTEJA A PARTE INTERNA disso. /
PROTEJA O EXTERIOR disso. / PROTEJA A PARTE DE FORA disso. /
PROTEJA A PARTE EXTERNA disso.

Onde disso podem ser caixas, elipses, polígonos ou caixas arredondadas. Perceba, no entanto, que a infelizmente só conseguimos usar uma "fita boa" em caixas. Em todos os outros lugares só conseguimos usar uma fita ruim, então não espere perfeição com nada a não ser caixas. Qualquer fita que você aplica permanece aplicada, então depois você provavelmente vai querer usar um desses comandos:

DESPROTEJA O INTERIOR disso. / DESPROTEJA A PARTE DE DENTRO disso. /
DESPROTEJA A PARTE INTERNA disso. / REMOVA A PROTEÇÃO DO INTERIOR disso. /
REMOVA A PROTEÇÃO DA PARTE DE DENTRO disso. / REMOVA A PROTEÇÃO DA PARTE INTERNA disso. /
DESPROTEJA O EXTERIOR disso. / DESPROTEJA A PARTE DE FORA disso. /
DESPROTEJA A PARTE EXTERNA disso. / REMOVA A PROTEÇÃO DO EXTERIOR disso. /
REMOVA A PROTEÇÃO DA PARTE DE FORA disso. / REMOVA A PROTEÇÃO DA PARTE EXTERNA disso.

Ou para agilizar o serviço:

DESPROTEJA TUDO. / REMOVA A PROTEÇÃO DE TUDO.

para começar do zero. Para conveniência, você pode remover toda a fita existente e colocar uma nova fita ao mesmo tempo com instruções como as seguintes. Acredite se quiser, estas são as que são mais frequentemente usadas:

PROTEJA SOMENTE O INTERIOR disso. / PROTEJA SOMENTE A PARTE DE DENTRO disso.
PROTEJA SOMENTE A PARTE INTERNA disso.
PROTEJA SOMENTE O EXTERIOR disso. / PROTEJA SOMENTE A PARTE DE FORA disso.
PROTEJA SOMENTE A PARTE EXTERNA disso.

Note que se você estiver desenhando e nada estiver aparecendo, isso provavelmente está acontecendo porque você está usando a fita crepe onde não quer, ou você esqueceu de ATUALIZAR A TELA conforme descrito no tópico Desenhar.

NOMES

Ao contrário dos compiladores de era neandertal, as regras para nomes são amplas e flexíveis.

Em geral, um nome pode ser uma palavra ou vároas, e pode começar com letras, dígitos e qualquer símbolo que eu não sejam sinais de pontuação, como um ponto, uma vírgula, um ponto e vírgula ou dois pontos.
Um nome é geralmente um substantivo, ou um substantivo com um ou mais adjetivos.
Você não deve usar artigos, verbos, conjunções ou preposições em nomes.

Nomes de tipo geralmente contém de uma a três palavras. Como byte ou nome do arquivo.

Os nomes dos campos geralmente são apenas um nome de tipo. Como número ou texto. Mas eles também podem incluir adjetivos como número total ou texto do primeiro nome. A parte do adjetivo pode ser usada como um apelido se não causar ambiguidade.

Nomes globais são frequentemente um tipo seguido por um adjetivo: a tecla shift.

Nomes de parâmetros parecem nomes de campo. Um tipo, com ou sem adjetivos. Uma caixa, por exemplo, ou uma cor de borda. Também neste caso o apelido funciona.

Os nomes de procedimentos iniciam com um verbo. Em seguida, uma série de parâmetros (com artigos indefinidos), frases e talvez um qualificador no final. Tal como remova o último byte de um texto ou centralize um ponto em uma caixa (horizontalmente).

Nomes de função sempre começam com PONHA e terminam com EM/DENTRO e um nome de tipo. Com uma frase possessiva no meio. Como coloque a linha do topo que faz parte da caixa em uma linha.

Nomes de decisores parecem nomes de procedimentos, exceto pelo fato do verbo normalmente aparecer em algum lugar no meio. Como em, um número É menor que outro número.

Os nomes de variáveis locais seguem o padrão de parâmetro. O compilador cria uma variável local sempre que encontra um nome com um artigo indefinido na frente de todo o corpo de uma rotina.

NÚMEROS ALEATÓRIOS

"Para fazer um sorteio são lançados os dados, mas quem determina o reultado é o Senhor." Provérbios 16:33 Então, eu acho que teremos que desistir dessa ideia de números aleatórios e nos contentar com números pseudo-aleatórios. Que é o que esse consegue gerar (não só esse, mas a maioria dos compiladores).

Na verdade, o compilador gera a mesma sequência de números "aleatórios" toda vez, a não ser que você semeie o gerador de números aleatórios com um valor inicial diferente (esse comportamento também se repete em outros compiladores).
Enfim, para semear o gerador de números aleatórios, use o comando:

```
SEMEIE O GERADOR DE NÚMERO ALEATÓRIOS.
```

Mas não pense que você vai sempre ter certeza do resultado. Ninguém sabe. Exceto o Senhor, é claro.

A rotina de números aleatórios mais básica é esta:

```
ESCOLHA um número.
```

O que retorna um número entre 0 e 2147483647. Você também pode fazer assim:

```
ESCOLHA um número ENTRE o número mínimo E o número máximo.
ESCOLHA um número ENTRE uma quantidade DE outro número.
(Por exemplo: Escolha um número entre 6 de 60).
```

Você também pode usar a aleatoriedade ao escolher posições na tela:

```
ESCOLHA um lugar em QUALQUER LUGAR dentro de uma caixa.
ESCOLHA uma localização NO RAIO DE uma distância DE outra localização.
```

Rotinas como essa estão sempre sendo adicionadas e aperfeiçoadas então é sempre bom dar uma olhada no código fonte do compilador. Basta procurar por "escolha" e você provavelmente encontrará a maioria deles.

E se você não tiver certeza qual usar, tente o cara ou coroa.

PALAVRAS-CHAVE

A maioria das linguagens de programação tem longas listas de palavras chave anômalas, cabalistas, enigmáticas, inescrutáveis, ofuscantes, "reservadas", tais como:

ABSTRACT, PROTECTED, SYNCHRONIZED, TRANSIENT, e VOLATILE.

Essa linguagem funciona de forma diferente. As palavras-chave são termos comuns. Artigos como:

UM, UMA, UNS, UMAS, O, A, OS, AS, etc.

Verbos usados frequentemente:

SER, ESTAR, PODER, FAZER, DEVER, POSSUIR e TER.

Algumas conjunções:

E, AMBOS(AS), MAS, QUALQUER, QUAISQUER, NENHUM(A), NEM e OU.

E muitas preposições:

EM CIMA, COMO, EM, ANTES, ENTRE, VIA, EMBAIXO, POR, DE e várias outras.

Alguns operadores aritméticos:

MAIS, MENOS, VEZES, DIVIDIDO POR, EM SEGUIDA, NULO, SIM, NÃO, CHAMADO e IGUAL.

Por último, palavras negativas:

NÃO, NEM, NENHUM, NADA, NUNCA, JAMAIS e similares.

Espero não ter esquecido nada importante. Na dúvida, consulte o código fonte para exemplos de uso.

PARÂMETROS

Uma variável se torna um "parâmetro" quando ela é passada para uma rotina. Para utilizar os parâmetros em uma rotina, você precisa informar a quantidade e os tipos de parâmetros no cabeçalho da rotina. Eis alguns cabeçalhos de rotina de amostra:

Para adicionar um número a outro número:
Para decidir se uma localização está dentro de alguns polígonos:
Para colocar o centro de uma elipse em uma localização:

A primeira rotina é um procedimento que espera dois parâmetros: um número e outro número. O primeiro é um parâmetro entrada; o segundo também é um parâmetro de entrada mas será devolvido na saída da rotina.

A segunda rotina é apenas um decisor comum. Ela também espera dois parâmetros, uma localização e alguns polígonos. Ambos os parâmetros são somente de entrada.

A terceira rotina é uma função com dois parâmetros: uma elipse e uma localização.
A elipse é o parâmetro de entrada e a localização é o parâmetro de saída.

As definições de parâmetros são fáceis de identificar porque sempre começam com um artigo indefinido (UM, UMA, UNS, UMAS ou ALGUM, ALGUMA, ALGUNS, ALGUMAS) seguido por um substantivo. Você pode ler mais sobre nomes sob o tópico Nomes.

Observe que quando parâmetros são passados para rotinas, o compilador passa o parâmetro original, e não cópias. É por isso que você pode usá-las na entradas, saída ou nos dois. Às vezes, no entanto, você queira manipular o valor de um parâmetro sem afetar a variavel original. Neste caso, você pode utilizar o seguinte comando:

PRIVATIZE um parâmetro.

E o compilador fará uma cópia do parâmetro para você. Conforme explicado anteriormente, você continuará utilizando o mesmo nome do parâmetro, já que a variável original será renomeada apenas temporariamente. A variável original receberá o sufixo original no nome dela, para que você ainda possa acessá-la se precisar.

POLÍGONOS

Existem duas definições importantes sobre polígonos no compilador:

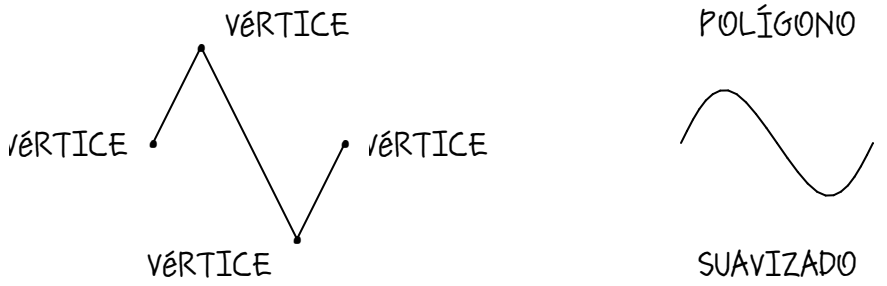
Um polígono é uma coisa com alguns vértices.
Um vértice é uma coisa com uma coordenada x, uma coordenada y, e uma localização na coordenada x.

Polígonos e vértices são "coisas" e, portanto, ao contrário dos outros objetos gráficos, eles têm que ser criados e destruídos. Você também precisa acrescentar os seus vértices aos seus polígonos. Exemplos:

CRIE um polígono.
CRIE um vértice A PARTIR DE uma localização.
ACRESCENTE um vértice A um polígono.
DESTRUA um polígono.

Ao usar o comando DESTRUA um polígogo, o compilador se livra tanto do polígono quanto dos vértices associados a ele.

Uma vez que você criar um polígono, você poderá DESENHÁ-lo. Você também pode realizar todas as habituais Transformações Gráficas nele. Você também pode usar o comando SUAVIZAR, e o compilador vai tentar arredondar o seu polígono, transformando cantos em curvas. Aqui está um polígono de amostra, com a versão normal e a suavizada:



Muito bem! Dessa forma fica fácil desenhar ondas sinoidais e outras formas geométricas sem usar um número real sequer. Eu gostaria que o Mestre Leopold Kronecker estivesse aqui pra ver isso.

POSSESSIVOS

Os Possessivos são normalmente usados para acessar campos nos registros. Conforme o exemplo abaixo:

nome do campo QUE FAZ PARTE DO nome do registro.

Mas eles também podem ser usados para fazer referência a uma função:

nome da função QUE FAZ PARTE DO nome da coisa.

Se o compilador não conseguir encontrar o campo dentro do primeiro nível registro, ele verifica se existe algum campo do registro que também seja um registro. E se encontrar, usa esse campo.

Mas a primeira coisa o compilador faz ao encontrar um possessivo é verificar a existência de três termos específicos.
O primeiro é o seguinte:

um ALVO que faz parte de um ponteiro

Este forma é usado apenas com ponteiros. Isso diz que você quer saber para que item o ponteiro aponta. "Um ponteiro de byte", por exemplo, refere-se ao endereço de um byte. "O alvo que faz parte do ponteiro de byte" se refere aos dados no byte.

Os outros possessivos especiais retornam "meta-dados" — dados a respeito dos dados. Um deles lhe traz o tamanho, em bytes, e o outro lhe traz o endereço:

[O campo] MAGNITUDE do registro x.
[O campo] PARADEIRO do registro y.

Você provavelmente não precisará destes com muita frequência, por isso que esses nomes esquisitos foram escolhidos.

PROCEDIMENTOS

Um "procedimento" é uma rotina que faz algo para você. Eis alguns cabeçalhos de procedimento de amostra:

Para converter um número em um texto:
Para centralizar uma caixa dentro de outra caixa (horizontalmente):
Para ocultar o cursor:

O formato geral é:

PARA fazer alguma coisa:

Os cabeçalhos de procedimento sempre começam com a palavra PARA, e sempre terminam com de dois-pontos. O alguma coisa entre elas segue as regras normais para nomes de rotina.

O conteúdo do corpo dos procedimentos são compostos por declarações: condicionais e imperativas, incluindo os imperativos internos como PRIVATIZE, PERCORRA, REPITA, PARE, e SAIA. No entanto, não se pode utilizar os comandos DIGA SIM e DIGA NÃO num procedimento.

O primeiro cabeçalho de amostra acima inclui um verbo, uma preposição e dois parâmetros. O verbo é converter e a preposição é em. Os parâmetros são um número e um texto.

O segundo procedimento é parecido com o primeiro, mas existe um qualificador: "(horizontalmente)".

A terceira rotina é um verbo seguido de um substantivo: o cursor.

Quando um substantivo estiver acompanhado de um artigo definido em um cabeçalho de rotina, ele geralmente será interpretado como uma variável global única, como por exemplo o comando desenhe a barra que está dentro do arquivo ambiente de trabalho. Mas eles também podem ser utilizados para se referir a uma pseudo-variável que não está definida precisamente no seu código. Como o cursor no exemplo acima, ou o último byte na rotina remova o último byte de um texto.

LOCALIZAÇÕES

Uma "localização" ou "posição" ou ainda "ponto na tela" é um dos objetos gráficos mais básicos do compilador. Esta não é bem a melhor definição pra agora vai servir:

Uma localização é um registro que tem uma coordenada x e uma coordenada y.

Esta é uma imagem de uma localização, com as partes acima identificadas. Note que estou usando os apelidos dos campos aqui, como você provavelmente usará em seu programa.

Localizações são feitas a partir de um x e um y, ou você pode obter uma de outro lugar:

FAÇA uma localização USANDO isto E aquilo.
ARMAZENE a posição do mouse EM uma localização.

O compilador consegue DESENHAR uma localização. Mas não espere que seja rápido o suficiente para ser útil. O processamento de vídeo do Windows é uma das suas piores características. E já que ele só oferece recursos ruins, isso não é lá muito encorajador.

Posições são usadas principalmente como componentes de outros objetos gráficos. Como caixas, linhas, vértices e polígonos. Às vezes, elas são usados como coordenadas abstratas sem representação visível, como "a localização do mouse" no exemplo acima. Consulte a página "Unidades de medida" para uma discussão completa sobre coordenadas.

O compilador possui rotinas que podem identificar se uma localização ESTÁ DENTRO ou NA BORDA de qualquer outro objeto gráfico (na borda significa em cima). Quando você vê se algo está DENTRO, isso inclui as bordas. As rotinas que identificam se algo está na borda são usadas pelo caderno e incluem uma margem de erro de 3 pixels para ficar mais fácil de clicar nas formas. Você pode copiar essas rotinas e retirar essas margens de tolerância, se quiser.

REGISTROS

Um record é uma coleção de itens que contém dados intimamente relacionados. Cada item é considerado um "campo". Os campos são descritos em sua própria página. Mas aqui estão alguns exemplos de registros:

Uma caixa tem
uma coordenada esquerda, uma coordenada superior, um coordenada direita, uma coordenada inferior, um canto superior esquerdo na coordenada esquerda, e um canto inferior direito na coordenada direita.

Uma caixa arredondada é uma caixa que tem
uma coordenada esquerda, uma coordenada superior, um coordenada direita, uma coordenada inferior, um canto superior esquerdo na coordenada esquerda, e um canto inferior direito na coordenada direita e um raio.

Um polígono é uma coisa com alguns vértices.

O primeiro registro de amostra, caixa, tem seis campos. Mas os dois últimos são na verdade "reinterpretações" dos primeiros quatro. Este tipo de coisa só funciona, é claro, quando as estruturas físicas de dados correspondem. Observe que a palavra tem é uma abreviação do termo é um registro com, que também pode ser usado.

O segundo registro, caixa arredondada, é uma extensão da caixa. Tem os mesmos campos que uma caixa, e mais um novo campo chamado de raio. É compatível com a caixa, e é possível usar todas as rotinas que funcionam nas caixas normais para manipulár as caixas arredondadas — a menos que uma rotina específica para caixas arredondadas tenha sido criada.

O terceiro registro, polígono, não tem nada além de uma lista de vértices. Como o polígono é definido como uma coisa, o compilador considera o polígono como uma estrutura dinâmica de dados (ao invés de uma estrutura estática). Isto significa que você é responsável por alocar e lidar com a memória usada por ele. Consulte o tópico Gerenciamento de Memória e a página sobre Polígonos para obter mais informações.

ROTINAS

Uma rotina é um pedaço de código que manipula uma ou mais variáveis em algumas formas bem definidas. As variáveis passadas para uma rotina são chamadas de "parâmetros", podendo ser apenas parâmetros de entrada, parâmetros de saída ou ambos. As variáveis definidas dentro de uma rotina são chamadas de "variáveis locaos" e não podem ser vistas fora da rotina (a menos que sejam passados como parâmetros). As variáveis que são acessíveis a todas as rotinas são chamadas de "variáveis globais".

Cada rotina tem duas partes, cabeçalho e corpo. O cabeçalho diz o que a rotina faz e define os parâmetros com os quais ele funciona. O corpo contém uma ou mais afirmações que fazem a rotina realmente funcionar. Declarações podem ser "condicionais" ou "imperativas". Existem três tipos de rotinas.

Um "procedimento" é uma rotina que simplesmente faz algo — um procedimento pode ser longo ou curto, grande ou pequeno, fácil ou difícil. Os cabeçalhos de procedimento sempre se parecem com isto:

PARA fazer alguma coisa:

Um "decisor" é uma rotina que diz "sim" ou "não" sobre algo, geralmente depois de examinar os parâmetros passados para ele. Um cabeçalhos de decisão é assim:

PARA DECIDIR SE alguma coisa:

Uma "função" é uma rotina que extrai, calcula ou deriva algo usando uma variável. Cabeçalhos de função assumem este formato:

PARA POR algo QUE FAZ PARTE DE algo EM uma variável temporária:

Ao contrário dos procedimentos e dos decisores, as funções não são normalmente chamadas directamente.
Em vez disso, o "algo que faz parte de algo" é utilizado como se fosse um campo em um registro.
Como um "centro que está dentro da caixa", que você não encontrará no registro "caixa", porque ele é calculado por uma função automaticamente.

SONS

Você pode reproduzir sons usando comandos como este:

REPRODUZA um arquivo wave.
TOQUE um arquivo wave E AGUARDE.

O "arquivo wave" deve estar no formato ".wav". Se você tocar e não utilizar o comando aguarde/espere, o seu programa continuará sendo executado enquanto o som é reproduzido. Se você utilizar o comando aguarde/espere, o fluxo de execução do seu programa vai parar até que o som tenha terminado de tocar.

Você também pode usar os seguintes comandos:

APITE.
CACAREJE (alerte o usuário).
ASSOVIE (sussurre).

O primeiro comando toca o som de erro padrão do Windows. O "cacarejo" é o som de notificação padrão do compilador, codificado em formato hexadecimal no compilador, basta pesquisar por "cluck". O terceiro som é o som de erro da CPU e não permite que o programa continue a ser executado até que termine de ser tocado, tornando-se a escolha ideal para testes. Consulte Depuração para obter mais informações.

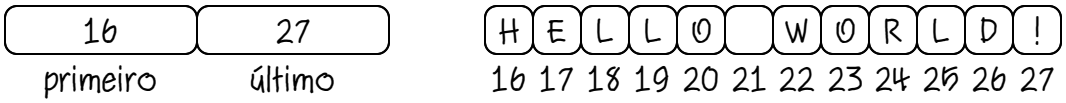
Você também pode falar no seu computador, com as trinta e nove funções esotéricas de "gerenciamento de fala" do Windows, ou usando um desses três comandos a seguir:

DIGA um texto.
DIGA um texto E AGUARDE.
AGUARDE ATÉ QUE A FALA TERMINE.

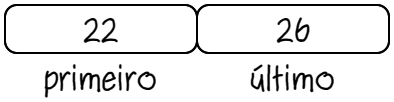
Para silenciar a conversa (mas não os outros sons), ative "a bandeira silenciosa".

SUBTEXTO

Uma "substring" ou subtexto faz parte de um texto. Subtextos são implementadoss através um registro embutido que se parece como um texto — um subtexto tem um par de ponteiros de bytes chamado primeiro e último — o que os torna compatíveis com o texto. Se, por exemplo, isso fosse um texto:



Isto poderia ser um subtexto (a parte "WORLD"):



Você pode usar o comando abaixo:

COLOQUE um subtexto EM um texto.

Para fazer com que o compilador defina o primeiro byte do subtexto e o último byte do subtexto abarquem completamente o texto. Isso permite que você consiga avançar ou retroceder dentro do texto ao adicionar algo ao primeiro byte ou removendo algum item do último byte. Você também pode usar o comando:

POSICIONE um subtexto EM um texto.

O que define o primeiro byte, mas não o último byte, do subtexto, deixando-o inicialmente em branco mas pronto para manipulação — ao adicionar o pultimo byte você pode "adicionar" o texto original no seu subtexto um byte por vez.

Procure por "subtexto" no código fonte para obter exemplos.

No entanto, o uso principal dos subtextos ocorre nos "percorredores", que são discutidos no neste glossário sob o tópico de mesmo nome.

TEMPORIZADORES

Um tique dura aproximadamente 1 milissegundo. "A contagem de tiques do sistema" é o número de milissegundos desde a última reinicialização. Ele reseta a cada 24,8 dias aproximadamente. O que acontece depois disso ninguém sabe,já que nenhum Windows ficou tanto tempo ligado sem dar pau.

Quando quiser, você pode:

AGUARDAR tantos milissegundos.

O compilador também entende unidades maiores como "minutos" ou "segundos".

O compilador possui uma variável chamada de temporizador que permite que você utilize os seguintes comandos:

ZERE o temporizador.
REINICIE um temporizador.
INICIE um temporizador.
PARE um temporizador.
REINICIE um temporizador.
INICIE um temporizador.
PARE um temporizador.

O compilador utiliza esses cronômetros para verificar o tempo de recompilação, por exemplo (que atualmente é cerca de 3 segundos, dependendo do hardware).
Procure a seção listagem para ver todos os temporizadores. Você pode usá-los para tornar seus programas mais rápidos.

Os tempos podem ser cronometrados simplesmente inserindo os comandos inicie um temporizador e pare o temporizador nos pontos apropriados no seu código.

Tempos cumulativos podem ser acumulados usando o comando zerar uma vez e reiniciar logo em seguida juntamente com o comando "parar".

Existe uma função no compilador que permite que você obtenha o texto de um temporizador a qualquer momento — mesmo durante sua execução. Você também pode utilizar as operações de concatenação de strings neles.

TERMOS

Um "termo" é uma referência a um pedaço de dados. Os termos são usados tanto em expressões quanto em instruções condicionais e imperativas para indicar o que deve ser operado. Os termos têm muitas variedades:

Um "termo literal" é um número, proporção, combinação, texto, ponteiro, bandeira ou um nibble literal. Consulte "Literais" para informações sobre como formular cada um deles.

Um "termo local" é uma variável definida dentro de uma rotina e que geralmente fica confinado dentro dela. Consulte "Variáveis Locais" para obter mais informações.

Um "termo global" é o nome de uma variável global. Consulte "Variáveis globais".

Um "termo sinalizado" (assinalado) é qualquer termo com um sinal de mais ou menos na frente dele. Um espaço é necessário após o sinal, para que o compilador não a confunda com a parte de um nome.

O "termo de proporção" é uma proporção feita a partir de outros termos, ao invés de números literais. Espaços são necessários ao redor da barra, como em "um valor de altura / um valor de largura".

Um "termo possessivo" é qualquer termo seguido por uma frase possessiva, como "o comprimento de dentro do texto" ou "magnitude que faz parte do polígono". Consulte a seção "Possessivos" para mais detalhes.

Um "termo coagido" é um termo cujo tipo você deseja mudar à força. O compilador sempre tratará um ponteiro, por exemplo, como um ponteiro — a menos que você coaga-o a ser outra coisa, como neste exemplo: "o ponteiro COMO UM NÚMERO". Normalmente você não vai precisar deste recurso, a menos que você seja uma pessoa que ama objetos e tenha definido um monte de coisas que são extensões de outras coisas.

Agora sei que isto parece complicado, e realmente é. Mas você não precisa pensar sobre nada disso, assim como você não precisa pensar em substantivos e verbos para falar português apropriadamente. Digite o que você está pensando e deixe o compilador tentar fazer o resto.

O CAMPO DE TEXTO

Existe um componente chamado "campo de texto". Ele é utilizado para implementar campos editáveis de texto, sejam eles grandes ou pequenos. Diálogos, por exemplo. Ou formas de texto em páginas. O editor de código, na verdade, é na maior parte apenas uma grande caixa de texto. Eis a definição:

Um campo de texto é algo com
uma caixa, uma origem,
uma cor de caneta, uma fonte, um alinhamento,
algumas linhas,
uma margem,
uma proporção de escala,
uma bandeira de embrulho,
uma bandeira de rolagem horizontal,
uma bandeira de rolagem vertical,
uma seleção,
uma bandeira modificada,
uma última operação,
alguns textos chamados desfazimentos, e
alguns textos chamados refazimentos.

Como você pode ver, esta não é uma definição trivial. Felizmente o compilador lida com a paior parte dos detalhes ra você. Normalmente, você não fará muito mais do que usar os seguintes comandos:

CRIE um campo de texto.
DESENHE um campo de texto.
DESTRUA um campo de texto.

Você deve inicializar a caixa do campo de texto, a caneta, a fonte, o alinhamento, a margem e as bandeiras depois de criá-lo. E você terá que passar todos os eventos relacionados à sua caixa de texto para o compilador, obviamente, para que ele possa cuidar de todas as coisas difíceis para você. Os manipuladores de eventos do campo de texto estão documentados nas duas páginas a seguir.

GERENCIADORES DE CAMPOS DE TEXTO

Conforme dito na página anterior, o compilador gerencia a maior parte dos eventos ou comandos que são enviados ao campo de texto, desde que você use os comandos apropriados:

GERENCIE um evento USANDO um campo de texto (tecla backspace).
GERENCIE um evento USANDO um campo de texto (tecla delete).
GERENCIE um evento USANDO um campo de texto (tecla seta para baixo).
GERENCIE um evento USANDO um campo de texto (tecla end).
GERENCIE um evento USANDO um campo de texto (tecla enter).
GERENCIE um evento USANDO um campo de texto (tecla esc).
GERENCIE um evento USANDO um campo de texto (tecla home).
GERENCIE um evento USANDO um campo de texto (clique duplo).
GERENCIE um evento USANDO um campo de texto (seta esquerda).
GERENCIE um evento USANDO um campo de texto (tecla page down).
GERENCIE um evento USANDO um campo de texto (tecla page up).
GERENCIE um evento USANDO um campo de texto (tecla imprimível).
GERENCIE um evento USANDO um campo de texto (seta direita).
GERENCIE um evento USANDO um campo de texto (tecla tab).
GERENCIE um evento USANDO um campo de texto (seta para cima).

Eu sei que pode parecer um incômodo despachar todos estes eventos separadamente, mas é exatamente isso o que torna o campo de texto geralmente útil. Por exemplo:

Se você tiver uma caixa de texto com apenas uma linha, provavelmente vai querer ignorar e as teclas de seta para cima e para baixo, enquanto que se você usar caixas de texto com várias linhas você vai querer utilizar essas teclas para mover o cursor de texto.

Se você estiver usando uma caixa de texto como uma caixa de diálogo, você provavelmente cancelará a operação quando a tecla Esc for pressionada, bem como você irá querer executar a operação ao pressionar a tecla Enter. Em um campo de texto normal, provavelmente você utilizará a tecla Enter para inserir uma quebra de linha por exemplo.

Entendeu o que eu quis dizer? Você está no comando. Portanto, é você que tem que emitir as ordens.

CONTINUAÇÃO

O compilador também gerencia de uma série de outras operações de texto de alto nível para você:

RECORTE O TEXTO DE um campo de texto.
COPIE O TEXTO DE um campo de texto.
COLE O TEXTO DE um campo de texto.

SELECIONE TODO O TEXTO DE um campo de texto.

GERENCIE A ALTURA DA FONTE usando um campo de texto e um valor de altura de fonte.
GERENCIE O NOME DA FONTE usando um campo de texto e um nome de fonte.

GERENCIE A COR DO TEXTO usando um campo de texto e uma cor.

GERENCIE O RECUO DO TEXTO usando um campo de texto.
GERENCIE O AVANÇO DO TEXTO usando um campo de texto.

TRANSFORME O TEXTO DE um campo de texto EM LETRAS MAIÚSCULAS.
TRANSFORME O TEXTO DE um campo de texto EM LETRAS MINÚSCULAS.

Ou, se você preferir:

DESFAÇA A OPERAÇÃO EM um campo de texto.
REFAÇA A OPERAÇÃO EM um campo de texto.

Você sempre pode usar os comandos acima. O máximo de vezes que você pode desfazer ou refazer uma operação é de 32.

Se você quiser ter uma noção boa de como funciona o campo de texto, eu sugiro que você experimente o "console" por um tempo. e então confira o código fonte do console. Depois disso, você pode querer dar uma olhada no editor de código. Mas se você realmente quer ver o campo texto em ação, dê uma olhada no caderno. Finalmente, gaste alguns minutos com as caixas de diálogo no ambiente de trabalho.

TIPOS

Um "tipo" é uma categoria ou espécie de coisa — um substantivo. Uma "instância" é uma coisa real de um tipo específico — um substantivo próprio. Globais, locais e parâmetros são instâncias concretas de tipos abstratos. Isso é o que o compilador entende sobre tipos:

Em primeiro lugar, existem seis tipos "embutidos" primitivos no meu compilador: BYTE, WYRD, NUMBER, POINTER, FLAG e RECORD. Consulte "Tipos Integrados".

Em seguida, há "subconjunto de tipos" que representam algumas das instâncias de algum outro tipo. O compilador, por exemplo, inclui muitos tipos de subconjunto, como estes:

Um número é contador.
Um nome é um texto.

Em terceiro lugar, o compilador reconhece "tipos de unidade de medida". Assim ele pode converter um tipo de unidade em outro. Exemplos:

Um pé tem 12 polegadas.
Uma hora é 60 minutos.
O compilador também entende tipos de "registro".

Consulte a seção Registros para mais detalhes.

E não vamos nos esquecer dos "tipos de ponteiro", embora você raramente precise usá-los diretamente. O compilador sabe, por exemplo, que "um ponteiro de bytes é um ponteiro para um byte" e ele uso ponteiros de bytes para gerenciar suas strings. Consulte Strings, Substrings, Percorredores, e Possessivos para obter mais informações.

Por último, o compilador reconhece todos os tipos de "coisa". Existem várias no compilador, incluindo console, evento, imagem, polígono e vértice, todas elas são discutidas em outros lugares neste glossário. E também pode definir as suas próprias coisas. Consulte o tópico sobre Coisas, e tente se lembrar das "obras de arte" do programa de exemplo.

TIPOS PREDEFINIDOS/EMBUTIDOS

O compilador tabalha com 6 espécies de tipos de dados primitivos: BYTE, WYRD, NÚMERO, PONTEIRO, FLAG e RECORD.

Bytes. Bytes, como você já sabe, nada mais são do que 8 bits de dados binários sequenciais. O compilador trabalha no formato decimal, logo valor de um byte vai de 0 até 255. O compilador usa a tabela ASCII sempre que é necessário converter um byte para uma caractere.

Wyrds. As wyrds são necessárias para comunicação com o Windows/Processador. Eles têm 16 bits de comprimento sendo que o valor mínimo corresponde a -32768 e o valor máximo corresponde a +32767. Os bits dentro de cada byte são armazenados da esquerda para a direita, mas os bytes em si são armazenadas da direita pra esquerda. Infelizmente essa é uma das coisas que não é possível alterar.

Números. O compilador trabalha números inteiros. Ou seja, números positivos e negativos. Eles têm 32 bits de comprimento com o valor mínimo sendo -2147483648 e o valor máximo sendo +2147483647. São armazenados da direita para a esquerda.

Pointers. OS endereços de memória são armazenados em ponteiros de 32 bits de comprimento, da direita para a esquerda. Os ponteiros possuem o mesmo intervalo que os NÚMEROS inteiros, mas os números negativos são reservados para uso do Windows.
O endereço 0 é inválido e é chamado de NIL. Para fazer com que um ponteiro fique vazio (NIL) é só usar o comando ESVAZIAR.

Flags. Apesar de ocuparem 32 bits na memória apenas o bit mais significativo da direita é utilizado. Na verdade, são os 8 primeiros bits começando a partir da esquerda, mas na prática é como se fosse o que eu falei antes. Dessa forma, o valor 0 é interpretado como "não" ou "falso" e 1 como "sim" ou "verdadeiro". Qualquer outro valor é inválido. Você pode LIMPAR uma flag para indicar "não" ou DEFINIR uma flag para indicar "sim".

Records. O último dos meus tipos embutidos é o record. O protótipo inicial ocupa zero bits na memória, mas você pode definir records de qualquer comprimento adicionando "campos" para o registro protótipo. Estes campos podem ser baseados em qualquer um dos tipos primitivos, incluindo outros records que você tenha definido anteriormente.

FONTES

No Windows, uma fonte é definida com quatorze parâmetros distintos. Complicado demais. O compilador usa uma definição bem mais simplicada:

Uma fonte tem um nome e uma altura.

O nome da fonte é o nome real armazenado em um arquivo de fonte. Ele pode ou não ser o mesmo que o nome do arquivo. Você provavelmente está familiarizado com nomes de fonte como Arial, Times New Roman e Courier New.

A altura da fonte pode ser especificada em qualquer unidade de medida conveniente. O caderno utiliza valores como 1/4 de polegada para poder se harmonizar com a função de alinhamento yank.

A fonte padrão" do compilador é chamada de Osmosian como homenagem aos criadores do projeto. O código hexadecimal da fonte está presente no noodle. O CAL "instala" a fonte na inicialização e deleta no encerramento. Ela mede 1/4 de polegada. Você está vendo uma amostra neste momento.

Para configurar uma fonte, basta utilizar o comando:

COLOQUE um nome E uma altura EM uma fonte.

Então, quando você estiver exibindo os itens na dela, informe que quer usar sua fonte:

Escreva "Olá, mundo!" no centro da caixa de seleção usando a fonte.

Se as suas fontes apareceram diferente do esperado, você provavelmente errou o nome da fonte.

Lembre-se, um nome de fonte não é necessariamente o nome do arquivo na pasta "fonte" do Windows. Em vez disso, você deve usar o nome do typeface exibido na caixa de visualização que aparece quando você clica duas vezes em um desses arquivos de fontes.

TRANSFORMAÇÕES GRÁFICAS

O compilador possui uma série de rotinas para manipulação gráfica. Os objetos gráficos podem ser manipulados de diversas formas. Por exemplo, você pode usar os comandos:

MOVA alguma coisa PARA CIMA tantos pixels.
MOVA alguma coisa PARA BAIXO tantos pixels.
MOVA alguma coisa PARA A ESQUERDA tantos pixels.
MOVA alguma coisa PARA A DIREITA tantos pixels.
MOVA alguma coisa USANDO um valor x E um valor y.
MOVA algo PARA um ponto na tela.

O último comando MOVA usa o canto superior esquerdo do ponto informado para efetuar o alinhamento do objeto. O compilador também aceita os seguinte comandos:

CENTRALIZE algo EM um ponto na tela.
CENTRALIZE algo EM uma caixa.

Álém dos comandos de posicionamento, existem comandos de transformação de objetos. Por exemplo:

INVERTA algo.
ESPELHE algo.
ROTACIONE algo.

O comando INVERTA inverte objetos verticalmente. O comando ESPELHE faz a mesma coisa, mas na horizontal. No momento o comando ROTACIONE só consegue rotacionar objetos em intervalos de 90 graus. Além disso o comando não funciona com textos. Essa á uma rotina que ainda está sendo aprimorada.

As últimas rotinas de transformação de objetos gráficos são:

REDIMENSIONE algo USANDO uma proporção.
REDIMENSIONE algo USANDO uma porcentagem.

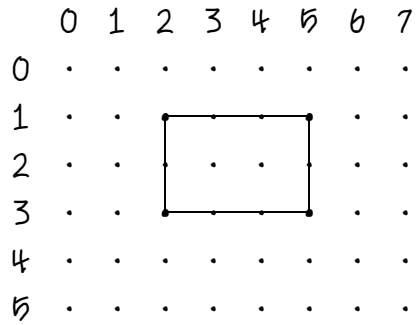
Consulte os tópicos Desenhando e Máscara para informações adicionais.

UNIDADES DA MEDIDA

A unidade de medida básica para objetos gráficos é a "twip", que é 1/20 de um ponto de impressora, ou 1/1440 de uma polegada. Todas as coordenadas são expressas diretamente em twips, ou são convertidas para twips. Eis a definição:

Um coordenada é alguns twips.

Agora, ao contrário do modelo matemático — que considera coordenadas como invisíveis abstrações indimensionadas — o modelo do compilador entende coordenadas como pontos reais em um dispositivo, como uma tela ou uma página. Valores grande de x e y vão além da tela ou página. Aqui, por exemplo temos uma caixa com o canto superior esquerdo em 2-1 e inferior direito, em 5-3:



Conte os pontos e note que a largura desta caixa é de quatro, não três, unidades. E que são três, e não duas unidades de altura. Medir desta forma faz com que o compilador desenhe bem — outra caixa entre 5-1 e 6-3, por exemplo, irá sobrepor corretamente esta caixa ao longo de sua borda esquerda. No entanto, você pode obter a largura e a altura desta caixa na forma "matemática" — que correspondem, cada uma, a uma unidade a menos, e são inúteis para desenho. Use as funções X-EXTENT e Y-EXTENT.

Outras unidades de medida que você vai encontrar são: milissegundos, segundos, minutos e horas; polegadas e pés; kilobytes, megabytes, e gigabytes; e "porcento", que geralmente é convertido para uma proporção com 100 no denominador.

VARIÁVEIS GLOBAIS

Uma variável "global" é uma variável que é visível para todas as rotinas, e que portanto pode ser usada por qualquer rotina em um programa. As variáveis globais podem ser definidos através de várias maneiras, mas suas definições sempre começam com um artigo definido (O, A, OS, AS). Eis alguns exemplos:

A seta do mouse é um cursor.
A tecla backspace é uma tecla igual a 8.
O maior número é [um número igual a] 2147483647.
A seta do mouse será inicializado com o valor 0 (indefinido).

A tecla backspace foi inicializada com um tipo explícito (tecla) e com o valor 8. A terceira variável usa um tipo implícito (colocado em colchetes como observação). Portanto, as formas gerais são:

O nome da variável aqui É UM nome de tipo da variável aqui.
O nome da variável aqui É UM nome de tipo da variável aqui IGUAL A valor da variável aqui.
O nome da variável aqui É valor da variável aqui.

Uma variável global "única" é um tipo especial de variável global que inclui a definição de de um tipo dentro dela. Exemplo:

O mouse tem uma tecla chamada botão esquerdo e uma tecla chamada botão direito.
Como só existe um mouse, não há necessidade de definir um registro chamado mouse. Em vez disso, a variável global e seu tipo podem ser definidos em uma única linha.
Há duas formas que podem ser utilizadas para definir uma variável global de tipo único. A primeira cria um novo tipo de registro, enquanto a segunda estende um registro já existente:

O insira o nome da variável aqui TEM insira os campos aqui.
O insira o nome da variável aqui É UM insira o tipo da variável aqui COM insira os campos aqui.

Uma variável global única é algo raro. Provavelmente porque são únicas.
Mas respondem à velha pergunta "Quem veio primeiro, o ovo ou a galinha?". A galinha, claro.

VARIÁVEIS LOCAIS

Uma "variável local" é uma variável que é propriedade privada de uma rotina. Variáveis locais não podem ser vistas ou modificadas por qualquer outra rotina. A menos que, claro, você as passe para outras rotinas como parâmetros.

O compilador cria uma nova cópia de cada variável local, a cada vez que uma rotina é chamada. A variável é inicializada com o valor 0. O que significa que uma rotina pode chamar a si mesa e tudo ainda vai funcionar. Isto é chamado de "recursão", e se você não sabe o que isso significa, não precisa disso. O compilador se lira das variáveis locais à medida que cada rotina é completada, para que elas não ocupem memória à toa.

Você cria uma nova variável local em uma rotina sempre que usa um artigo indefinido (A, AN, ANOTHER ou SOME) em um comando. Por exemplo:

```
Coloque a localização do mouse em outra localização.
Coloque a coordenada esquerda da tela em uma coordenada esquerda da caixa.
Coloque 101 em outro número de curso.
```

No primeiro exemplo, a frase uma localização faz com que eu faça uma nova variável local chamada "a localização". Em seguida, o compilador põe a localização atual do mouse na variável.

O segundo exemplo coloca a coordenada esquerda da tela em uma nova coordenada esquerda da caixa. O restante das coordenadas da caixa — superior, direita e inferior — estão definidas como zero.

O terceiro exemplo coloca um literal 101 em uma nova variável local do tipo número. Esta variável é definida com adjetivos anteriores ao nome do tipo, então ela pode ser referenciada pelo seu nome completo, O outro número de curso, ou por apelido, o outro curso. Você pode ler mais sobre nomes sob o tópico Nomes.

Consulte também a página Laços, onde uma variável local e um decisor nos permite fazer laços contados sem adicionar novas palavras-chave ao compilador.

Índice temático

ÍNDICE TEMÁTICO