Introdução à Programação C

Fabio Mascarenhas - 2014.2

http://www.dcc.ufrj.br/~fabiom/introc

Estado

- A interação com o usuário de nossas aplicações iniciais é bem linear: entrada com `scanf`, saída com `puts` e `printf`
- Assim conseguimos manter todo o *estado* da aplicação dentro da função `main`, passando para as funções auxiliares a parte que cada uma precisa
- A passagem desse estado pode não ser *conveniente*, mas é sempre *possível*
- Como exemplo de uma aplicação com estado mais complicado, vamos modelar um jogo simples de forca com quatro letras

Forca de 4 letras

- O estado do nosso jogo tem três partes:
  - A palavra, formada por quatro caracteres
  - O que já foi revelado, formado por quatro booleanos, um para cada caractere
  - O número de erros do jogador
- Precisamos de funções auxiliares para fazer uma *rodada* do jogo: dada uma letra, ou atualizar o que já foi revelado ou sinalizar o erro, e verificar se o jogo terminou com vitória ou derrota do jogador

Variáveis globais

- Passar partes do estado para dentro e fora das funções auxiliares nem sempre é conveniente
- Por esse motivo, é comum aplicações usarem *variáveis globais* para guardar o seu estado
- Uma variável global é uma variável declarada no corpo do programa, normalmente antes das funções auxiliares
- Uma variável global pode ser lida ou escrita de qualquer função do programa
- Mudanças em uma função são vistas por outras

```
static char letra1 = ‘A’;
static char letra2 = ‘B’;
static char letra3 = ‘R’;
static char letra4 = ‘A’;
static int erros = 0;
```

Efeitos colaterais

- Vamos fazer uma nova versão do jogo de forca guardando o estado em variáveis globais
- Podemos diminuir as listas de parâmetros de nossas funções auxiliares, fazendo elas trabalharem diretamente com as funções globais
- Nossas funções auxiliares agora têm *efeitos colaterais*: além de processar sua entrada e retornar uma saída, elas também podem modificar o conteúdo de variáveis globais
- Isso não fica explícito na *assinatura* da função, então devemos adicionar um *comentário* para deixar claro quais são os efeitos de cada função

Aplicações gráficas

- Guardar estado em variáveis globais não é só uma conveniência para certas classes de aplicações

- Vamos trabalhar agora com aplicações gráficas interativas simples, envolvendo desenhos, sons, passagem do tempo (para animações) e interação com o teclado e o mouse

- O fluxo de controle nessas aplicações não é linear, entrando em uma função main e percorrendo ela até o final, mas sim baseado em *eventos*

- Cada tipo de evento tem uma função principal diferente, e a única maneira dessas funções cooperarem é através de estado global

Eventos

- Cada evento de nossas aplicações gráficas corresponde a uma função principal no programa, com as assinaturas abaixo:

```
void gui_init(String *titulo, int *largura, int *altura);
void gui_tecla(String tecla, int soltou);
void gui_tique(double dt);
void gui_desenhar();
```

- O evento `gui_init` indica a carga inicial da aplicação, e seus parâmetros de saída deixar ela dizer qual o título e dimensões da sua janela de desenho

- Os eventos `gui_tecla` e `gui_tique` indicam que uma tecla foi pressionada ou solta, ou que aconteceu um tique do relógio, tendo passado `dt` segundos desde o último

- Finalmente, `gui_desenhar` indica para a aplicação que ela deve desenhar um *quadro* de sua interface

Saída

- A aplicação tem um console onde ela pode fazer saída com `puts` e `printf`, mas ele deve ser usado apenas para indicar erros e depurar

- O mecanismo principal de saída são as funções de desenho, que devem ser usadas dentro do evento `gui_desenhar`:

```
void tela_ret(int x, int y, int larg, int alt, double r, double g, double b);
void tela_circ(int x, int y, int raio, double r, double g, double b);
void tela_elipse(int x, int y, int rx, int ry, double r, double g, double b);
void tela_triang(int x1, int y1, int x2, int y2, int x3, int y3,
        double r, double g, double b);
void tela_texto(int x, int y, String txt, double r, double g, double b);
void tela_letra(int x, int y, char c, double r, double g, double b);
```

- As cores para desenho são dadas pelos componentes vermelho, verde e azul, com valores entre 0 e 1

- A coordenada (0,0) é sempre o canto superior esquerdo da tela

Sons e encerramento da aplicação

- A aplicação também pode tocar sons que estejam gravados em algum arquivo `.wav`, dentro de eventos `gui_tecla`, `gui_tique` ou `gui_desenhar`:

```
void som_tocar(String arqwav);
```

- Para sinalizar que a aplicação deve ser encerrada, usamos a função `gui_quit`:

```
void gui_quit();
```

- Essa função não encerra a aplicação imediatamente; ela tem primeiro que terminar de desenhar o próximo quadro

- O tipo `String` usado nessas funções corresponde às cadeias de caracteres entre aspas duplas que temos usado

strcmp(s1, s2) == 0 se e somente se s1 igual a s2

Forca gráfica

- Podemos dar uma interface gráfica para nosso jogo de forca, aproveitando tanto o estado global quanto as funções auxiliares da versão anterior

- Essas funções serão o *modelo* da nossa aplicação

- Sempre que fazemos uma aplicação com uma interface gráfica é importante separar o modelo do código responsável pela interação com o usuário

- Idealmente, o mesmo modelo serviria tanto para uma aplicação com entrada e saída pelo console e para uma aplicação com entrada e saída por uma interface gráfica