navegação

- Página Inicial
- Equipe
- Atividades
- Localização
- Contatos
- Área de Membros
- Pastas das Disciplinas
  - EEL170 - Computação I
  - EEL270 - Computação II
  - EEL670 - Linguagens de Programação
  - EEL875 - Internet e Arquitetura TCP/IP
  - EEL878 - Redes de Computadores I
  - EEL879 - Redes de Computadores II
  - EEL480 - Sistemas Digitais
- Tutoriais
- Webmail
- Novidades
- Eventos

Aula Prática 08 - Roteiro

12/10/2019 - Roteiro referente à aula prática 08 - Algoritmo de Preenchimento de Áreas

Versão: 08/10/2019

Prazo: 15/10/2019 - 8:00

Observações:

- Leia este enunciado com **MUITA** atenção até o final antes de iniciar o trabalho.
- Este roteiro está disponível no formato PDF. Para acessá-lo, clique aqui.
- Os arquivos solicitados deverão estar disponíveis nos diretórios correspondentes (**Aulas-Praticas** e **RCS**) até o prazo estipulado acima. Cuidado com os nomes dos diretórios e dos arquivos. Deverão ser exatamente os definidos neste roteiro (maiúsculas, minúsculas, caracteres especiais e extensões, se existentes).
- As tarefas deverão ser executadas na ordem solicitada neste roteiro.
- A compilação e a *linkedição* deverão ser executadas utilizando-se tanto o *gcc*, quanto o *clang* . Em ambos os casos deverão ser utilizados os flags "*-Wall -std=c11*".
- Além disso, deverão ser executadas sem mensagens de advertência e sem mensagens de erro, tanto no *CentOS* 7.x, quanto no *FreeBSD* 11.x.
- No *CentOS* o comando *make* corresponde ao *GNU Make*, enquanto que no *FreeBSD* o comando é nativo. Estas duas variantes não são cem por cento compatíveis e por isso serão necessários dois arquivos de dependências, o *GNUmakefile* e o *BSDmakefile*. No *FreeBSD* o comando *gmake* poderia ser utilizado com o arquivo *GNUmakefile*, mas isto está fora do escopo desta aula.
- Inclua, sempre que necessário, o comando para criar uma cópia do binário com a identificação do sistema operacional e do *compilador/linkeditor* utilizados.
- Inclua, no início de todos os arquivos solicitados (*.c e *makefile), os seguintes comentários:

```
Universidade Federal do Rio de Janeiro
Escola Politecnica
Departamento de Eletronica e de Computacao
EEL270 - Computacao II - Turma 2019/2
Prof. Marcelo Luiz Drumond Lanza
Autor: <nome completo>
Descricao: <descrição sucinta dos objetivos do programa>

$Author$
$Date$
$Log$
```

Um dispositivo de saída (monitor) pode ser visto como uma matriz bidimensional de pixels. Para simplificar, cada pixel poderá estar apagado ou aceso. Considerando-se um terminal do tipo texto, um pixel será representado por um caractere, ou seja, '0' se estiver apagado e '1' se estiver aceso.

Um algoritmo de preenchimento de área utilizado neste ambiente recebe as informações atualizadas sobre todos os pixels do monitor, incluindo a definição de um polígono, ou seja, os pixels correspondentes aos lados do polígono deverão estar acesos. Além disso, o algoritmo deverá receber a quantidade de linhas e de colunas do monitor e as coordenadas de um pixel localizado dentro do polígono.

A partir destas informações, toda a área interna ao polígono deverá ser preenchida (os pixels deverão ser definidos pelo caractere '1').

Se o pixel inicial estiver fora do polígono toda a área extena ao polígono será preenchida (com exceção aos casos que dois lados paralelos estejam nas extremidades do monitor).

O primeiro passo no algoritmo é verificar se o pixel desejado já está aceso. Se o mesmo estiver aceso nenhuma ação deverá ser executada.

Caso contrário, o pixel em questão deverá ser aceso e o algoritmo deverá ser executado novamente para os quatro pixels vizinhos a este pixel (esquerda, direita, abaixo e acima do pixel em questão).

Nas questões a seguir, as macros NUMERO_MAXIMO_LINHAS e NUMERO_MAXIMO_COLUNAS representam os maiores valores permitidos pelo algoritmo. Embora estes valores sejam iguais a 250 e 800, um monitor de teste poderia ter por exemplo, 50 linhas e 100 coluinas, mas náo poderia ter 300 linhas e 400 colunas, já que o número máximo de linhas definido é 250.

1. Baseado no algoritmo acima, crie o arquivo "*aula0801.h*" contendo a definição das macros APAGADO ('0'), ACESO ('1'), NUMERO_MAXIMO_LINHAS (250) e NUMERO_MAXIMO_COLUNAS (800), do tipo *tipoErros* (tipo enumerado contendo os códigos de retorno da função *PreencherPoligono*, por exemplo, ok - valendo 0, abscissaInvalida - valendo 1, etc.), do tipo *tipoPixel* (enumerado contendo os valores apagado e aceso) e do protótipo da função *MostrarMonitor*. Esta função deverá receber uma matriz de pixels (tipoPixel) correspondendo ao monitor, além de suas dimensões e do tempo de congelamento da tela (em microsegundos). A função deverá exibir o conteúdo desta matriz. A macro referente à combinação *ifndef* e *define*, como por exemplo _AULA0801_, deverá ser definida como uma *string* valendo:

   "@(#)aula0801.h $Revision$"

   tipoErros
   MostrarMonitor (tipoPixel monitor [NUMERO_MAXIMO_LINHAS] [NUMERO_MAXIMO_COLUNAS], unsigned maximoLinhas, unsigned maximoColunas, useconds_t tempoEspera);

   Observação:
   O uso do operador ternário na chamada da função printf que exibe cada pixel (cada caractere correspondente) é obrigatório.

   Operador Ternário:
   Condição ? verdadeiro : falso

2. Crie o arquivo "*aula0801.c*" contendo o código fonte da função *MostrarMonitor*. Esta função deverá conter, antes da exibição do conteúdo, uma chamada para a função *system* executando o comando *clear*. Após a exibição do conteúdo, esta função deverá conter uma chamada para a função *usleep* (com o tempo recebido). A função deverá retornar *ok* (ZERO) ou o código de erro correspondente.
3. Inclua, no arquivo "*aula0801.h*", a definição do protótipo da função *LimparMonitor*. Esta função deverá receber as informações sobre todos os pixels do monitor em questão, além do número máximo de linhas e do número máximo de colunas deste dispositivo. Se todos os argumentos forem válidos, a função deverá "apagar" todos os pixels do dispositivo. A função deverá retornar *ok* (ZERO) ou o código de erro correspondente.

   tipoErros
   LimparMonitor (tipoPixel monitor [NUMERO_MAXIMO_LINHAS][NUMERO_MAXIMO_COLUNAS], unsigned maximoLinhas, unsigned maximoColunas, useconds_t tempoEspera);

4. Inclua, no arquivo "*aula0801.c*", a implementação da função *LimparMonitor*. A função *MostrarMonitor* deverá ser executada no final da execução da função *LimparMonitor*.
5. Inclua, no arquivo "*aula0801.h*", a definição do protótipo da função *DesenharRetangulo*. Esta função deverá receber as informações atualizadas sobre todos os pixels do monitor, o número máximo de linhas e o número máximo de colunas do dispositivo, além das coordenadas do canto superior (esquerdo ou direito) e do canto inferior (direito ou esquerdo) do retângulo desejado. Além disso, a função deverá receber o tempo de congelamento da tela (em microsegundos). Se todos os argumentos recebidos forem válidos, a função deverá "acender" todos os pixels que correspondam aos lados do retângulo em questão. A função deverá retornar *ok* ou o código de erro correspondente.

   tipoErros
   DesenharRetangulo (tipoPixel monitor [NUMERO_MAXIMO_LINHAS][NUMERO_MAXIMO_COLUNAS], unsigned maximoLinhas, unsigned maximoColunas, unsigned ordenadaCantoEsquerdo, unsigned abscissaCantoEsquerdo, unsigned ordenadaCantoDireito, unsigned abscissaCantoDireito, useconds_t tempoEspera);

6. Inclua, no arquivo de "*aula0801.c*", a implementação da função *DesenharRetângulo*. A função *MostrarMonitor* deverá ser executada no final da execução da função *DesenharRetangulo*.
7. Inclua, no arquivo "*aula0801.h*", a definição do protótipo da função *PreencherPoligono*. Esta função deverá receber as informações atualizadas sobre todos os pixels do monitor, incluindo a definição de um polígono, ou seja, os pixels correspondentes aos lados do polígono deverão estar acesos. Além disso, a função deverá receber a quantidade de linhas e de colunas do monitor e as coordenadas de um pixel localizado dentro ou fora do polígono. Esta função deverá receber também o tempo de congelamento da tela (em microsegundos). Se todos os argumentos forem válidos, o polígono deverá ser preenchido de acordo com o algoritmo definido acima.

   tipoErros
   PreencherPoligono (tipoPixel monitor [NUMERO_MAXIMO_LINHAS][NUMERO_MAXIMO_COLUNAS], unsigned maximoLinhas, unsigned maximoColunas, unsigned linha, unsigned coluna, useconds_t tempoEspera);

8. Inclua, no arquivo de "*aula0801.c*", a implementação da função *PreencherPoligono*. A função *MostrarMonitor* deverá ser executada toda vez que um pixel for modificado.
9. Crie o arquivo "*aula0802.c*" contendo a implementação do programa de testes da função *PreencherPoligono*. Este programa deverá receber o número de linhas e o número de colunas do dispositivo desejado, as coordenadas do canto superior esquerdo/direito e do canto inferior direito/esquerdo de um retângulo, as coordenadas de um ponto qualquer o tempo de congelamento da tela (em ms). Todos estes argumentos deverão ser recebidos via argumentos da CLI (argv's) e na ordem definida anteriormente. Se todos os argumentos recebidos forem válidos, o programa deverá chamar as funções necessárias (LimparMonitor, MostrarMonitor, DesenharRetangulo e PreencherPoligono). Lembre-se que as funções deverão ser testadas tanto com argumentos válidos, quanto com argumentos que gerem condições de erro na função.
10. Inclua, nos arquivos de dependências, a macro AULA0802OBJS - correspondendo aos arquivos necessários para criar o executável a partir dos arquivos "*aula0801.c*" e "*aula0802.c*". Além disso, defina a macro AULA08, equivalendo ao executável *"aula0802"*, e altere a macro EXECS, de forma que o valor da mesma inclua os executáveis criados na aula 06. Os arquivos de dependências deverão incluir ainda os objetivos *aula08* (executável da aula 08) e *aula0802* (executável criado a partir dos arquivos definidos pela macro AULA0802OBJS} com os comandos correspondentes.
11. Crie e teste as quatro versões do executável *aula0802*.
12. Submeta os arquivos *aula0801.h*, *aula0801.c*, *aula0802.c* e **makefile* ao sistema de controle de versão.
13. Recupere uma cópia de leitura dos arquivos *aula0801.h*, *aula0801.c* e *aula0802.c* e uma cópia de escrita dos arquivos **makefile*.

Outubro 2019

Do	Se	Te	Qu	Qu	Se	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

notícias

Horário 2017/2
31/07/2017

Estágio Supervisionado - Importante
05/03/2012

Mais notícias...