DCC003 – Algoritmos e Estruturas de Dados I
Trabalho Prático 1

Professor: Daniel Fernandes Macedo / Jefersson Alex dos Santos
Data de entrega: a definir em sala de aula
Trabalho em grupos de dois ou três alunos.
Valor do trabalho: 20% da nota dos exercícios práticos

1. Entrega do código

O código e a documentação devem ser entregues em um arquivo Zip (não pode ser RAR nem .tgz nem .tar.gz) no Prático, contendo um arquivo **readme.txt** com o nome dos integrantes, e um arquivo PDF da documentação. Incluam todos os arquivos (arquivos .c, e .h), que devem todos estar **em um único diretório**. Submissões onde os programas não seguem as especificações de parâmetros e nomes, arquivos necessários faltando **não serão corrigidos**. **Programas que não compilarem também não serão corrigidos.** Vocês poderão fazer o trabalho em qualquer editor (Code::Blocks, Visual C++, TextPad, etc).

2. Documentação

A documentação, além de ser entregue com o Zip junto ao código, deve ser entregue impressa ao professor até o dia seguinte à entrega, na sala de aula. Caso não tenhamos aula no dia seguinte ao prazo para envio, iremos combinar antecipadamente qual é a forma mais conveniente de entrega da documentação. Trabalhos que forem entregues no Prático mas sem a documentação impressa **não serão** corrigidos.
O texto da documentação deve ser breve, de forma que o corretor possa entender o que foi feito no código sem ter que entender linha a linha dos arquivos. Implementações modularizadas deverão mencionar quais funções são implementadas em cada módulo ou classe. A documentação deve conter os seguintes itens:

- Sumário do problema a ser tratado
- Uma descrição sucinta dos algoritmos, das principais funções, e procedimentos e as decisões de implementação
- Decisões de implementação que porventura estejam omissos na especificação.
- Testes que mostrem que a implementação é robusta e gera os resultados esperados e no formato indicado.

3. PhotoCommand

Neste trabalho vocês vão desenvolver o PhotoCommand, um editor gráfico de modo texto, que gera figuras do tipo bitmap. As figuras bitmap (aquelas com terminação .bmp, que os Windows antigos usavam muito) são figuras onde cada byte representa a cor de um pixel.

Vocês irão desenvolver um programa, em um arquivo com o nome "`photocommand.c`" que, dado um conjunto de comandos para a geração de uma ou mais imagens, gere na saída as imagens desejadas.

4. Formato da entrada e da saída

O PhotoCommand irá receber como entrada um conjunto de comandos, lidos a partir do teclado. Cada comando é definido como na tabela abaixo:

Comando	Funcionamento
I C L	Gera uma nova imagem com C colunas e L linhas.
Z	Zera a imagem, ou seja, gera uma imagem onde todos os pontos possuem a cor branca (cor `'O'`).
C X Y C	Colore o pixel (X,Y) com a cor C.
V X Y1 Y2 C	Gera uma linha vertical, partindo do ponto (X,Y1), e indo até o ponto (X,Y2). A cor da linha será C.
H X1 X2 Y C	Gera uma linha horizontal, partindo do ponto (X1,Y), e undo até o ponto (X2,Y). A cor da linha será C.
R X1 Y1 X2 Y2 C	Desenha um retângulo, de cor C, delimitado pelos pontos definidos por (X1,Y1) e (X2,Y2)
S NOME	Salva o arquivo (no nosso caso, imprime o arquivo na tela) com o nome dado. O nome tem um limite do tipo MSDOS: até 8 caracteres para o nome, até 3 caracteres para extensão de arquivo.
F	Fecha a sessão de edição.

Para simplificar a programação, assumam que a imagem pode ter no máximo 250 linhas ou 250 colunas. O seu programa deve ser "esperto", ou seja, deve ignorar comandos inválidos (comandos com letras diferentes das definidas acima). O seu programa não deve imprimir nenhuma mensagem de erro quando um comando inválido for digitado, somente ignorar a linha de entrada. Vocês podem assumir que os comandos, quando forem válidos, sempre estarão corretos (número correto de parâmetros, e valores de parâmetros que não ultrapassam a figura).

Um exemplo de entrada segue abaixo:

```
I 5 6
C 2 3 A
S one.bmp
G 2 3 J
R 1 1 5 6 J
V 2 3 4 W
H 3 4 2 Z
S two.bmp
F
```

A cada comando S NOME, imprima o nome do arquivo <NOME> e o conteúdo da imagem, **ambos na tela. Não é preciso efetivamente gravar em arquivos neste trabalho**. Como exemplo, a saída para a entrada acima seria a seguinte. **Atenção: não testem o seu programa somente com este conjunto de entradas e saídas, pois este teste é bem simples...**

```
one.bmp
OOOOO
```

```
OOOOO
OAOOO
OOOOO
OOOOO
OOOOO
two.bmp
JJJJJ
JJZZJ
JWJJJ
JWJJJ
JJJJJ
JJJJJ
```

Prestem atenção na forma como os dados serão lidos e impressos, pois os trabalhos serão corrigidos por um programa automático, como nas aulas práticas! Assim, alguns avisos:

- Programas que não compilarem serão zerados automaticamente;
- Programas que não aceitarem a entrada como formatado acima receberão nota zero;
- Programas que imprimirem saída diferente da que foi descrita acima também serão penalizados.
- Nomes de arquivo diferente do definido serão penalizados (não iremos zerar a nota, mas terá uma penalidade);
- Cópias de trabalho são inaceitáveis, podem ser detectadas automaticamente pelo nosso sistema de execução do código (comparamos com as entregas de outros colegas e também com trabalhos na Internet) e serão repassadas ao conselho disciplinar.

5. Desconto de nota por atraso

Os trabalhos poderão ser entregues até a meia-noite do dia especificado para a entrega. O horário de entrega deve respeitar o relógio do sistema XXXXXXXX, ou seja, a partir de 00:01 do dia seguinte à entrega no relógio do XXXXXXXX, os trabalhos já estarão sujeitos a penalidades. O valor do trabalho irá decrementar 5% a cada hora de atraso. Para um aluno que entregou o trabalho à 1:38, ou seja, com 1:38 de atraso, iremos descontar 10%, empregando a fórmula a seguir:

```
Nota = valor_correção * (1-0.05*horas_atraso)
```