Etapa 4 de Laboratórios de Informática II

Licenciatura de Engenharia Informática

Ano Lectivo 09/10

Versão 1.0

Conteúdo

1 Critérios de Aprendizagem

No final desta etapa os alunos deverão ser capazes de:

1. Organizar o código de forma estruturada;
2. Modificar funções prévias para as dotar de mais funcionalidades;

2 Enunciado

Implementar os seguintes comandos **sem qualquer tipo de variação na sintaxe**:

`vc`	Verificar cores; isto implica percorrer o tabuleiro e aplicar as regras das células brancas e pretas;
`tc`	Tentar cor; este comando recebe como argumentos dois números *col* e *lin* referindo-se ao número da coluna e linha de uma célula e deve atribuir-lhe uma cor se for impossível atribuir a cor contrária;

`tc` Tentar cor invocado sem argumentos; neste caso ele deve mudar a primeira célula que conseguir.

3 Verificação

Na etapa 3 criámos os automáticos que permitiam que os comandos se tornassem mais potentes. Mas o seu funcionamento não é retroactivo. Nesta etapa vamos criar um comando mais útil, chamado `vc`, porque é uma mistura do `ab`, `ap`, `vb` e `vp`. O que acontece é que para cada casa do tabuleiro com cor aplicamos a regra correspondente do jogo.

Para as brancas varremos a linha e a coluna e todas as letras iguais devem ser pintadas de preto, falhando caso elas tenham a cor contrária. Para as pretas colorimos as suas vizinhas ortogonais de branco, falhando caso elas tenham a cor oposta. Tal como nos casos anteriores o falhar é imprimir o erro `E_WRONG_SOLUTION`.

4 Tentativa

O comando `tc` é um comando mais inteligente que só podia surgir após a etapa 3. O que ele faz é tentar atribuir ambas as cores[1] à célula e depois verificar se ocorreu algum erro utilizando as três regras (a das brancas, a das pretas e a dos caminhos). Há três situações possíveis:

Ambas Neste caso não deve ocorrer nada;

Uma das cores Se só uma das cores puder ser atribuída e a outra não então o comando atribui essa cor à célula em causa;

Nenhuma Se a célula não pode ser nem branca nem preta então há erro (`E_WRONG_SOLUTION`).

Se o comando for invocado sem argumentos, ele deve procurar a primeira célula à qual possa aplicar este procedimento. Para isso o comando deve varrer as células linha a linha (num tabuleiro de tamanho n começa na primeira linha $(1,1),\ldots,(n,1)$, depois passa para a segunda $(1,2),\ldots,(n,2)$, até chegar à última linha $(1,n),\ldots,(n,n)$). Repare que tal como quando é invocado com argumentos, o comando só atribui cor a uma única célula[2].

5 Avaliação

Esta etapa vale no máximo 4 valores e deve ser entregue até ao fim do dia 16 de Maio no site da unidade curricular. Para não reprovar à unidade curricular

[1] branca e preta

[2] excepto é claro se `ab` e/ou `ap` estão ligados

os alunos precisam de tirar um mínimo de 50% da classificação desta etapa (2 valores). A entrega atrasada incorre numa penalização de 10% ao dia.

Esta avaliação de grupo está sujeita a um factor "avaliação individual" cujo valor é determinado durante as aulas de acompanhamento. Caso um aluno não apareça a essas avaliações individuais sem apresentar motivos de força maior devidamente justificados por documentos, reprova automaticamente à disciplina por faltar à avaliação.

5.1 Material a entregar para cada uma das etapas

O grupo deverá entregar em cada etapa um arquivo criado com o comando `tar` e comprimido com o algoritmo de compactação `bzip2` contendo a seguinte informação:

code Uma pasta com todo o código fonte do trabalho que deverá incluir a `makefile` utilizada para compilar todo o código e gerar a documentação;

doc Uma pasta com a documentação em formato `html` gerada a partir do código utilizando a ferramenta `Doxygen`;

relatorio.pdf Um ficheiro gerado utilizando o comando LaTeX no formato `pdf` com o relatório;

doc.pdf Um ficheiro gerado utilizando o comando LaTeX a partir dos ficheiros gerados pela ferramenta `Doxygen`.

O arquivo deverá utilizar o seguinte arquétipo para o nome:
`g<número do grupo><turno>-et<número da etapa>.tar.bz2`
Por exemplo, o ficheiro entregue na quarta etapa pelo grupo número 3 do turno PL1 teria como nome `g3PL1-et4.tar.bz2`.

5.2 Relatório

Cada etapa deve vir acompanhada de um relatório escrito em LaTeX e do código documentado. O relatório deve necessariamente ter as seguintes secções:

Resumo Onde se apresenta um breve resumo do relatório que não deverá ultrapassar as 250 palavras;

Introdução Onde se apresenta a introdução do relatório, tipicamente, deverá ter subsecções como motivação, objectivos e estrutura do relatório;

Desenvolvimento O desenvolvimento poderá ser mais do que uma secção, no caso deste relatório deverá explicar as opções tomadas e apresentar breves algoritmos que ajudem a compreender as partes críticas do código;

Conclusão Onde se apresentam as conclusões objectivas do trabalho efectuado;

Bibliografia Onde se citam as referências bibliográficas utilizadas no trabalho.

Lembre-se que o relatório pretende explicar quais foram as dificuldades encontradas e como estas foram resolvidas. A apresentação deverá ser técnica de forma a ajudar o avaliador a perceber o que foi feito e como foi feito.

É obrigatório apresentar a documentação gerada automáticamente através do código comentado utilizando o `Doxygen` `http://www.stack.nl/~dimitri/doxygen`. Todas as funções deverão ser documentadas com uma pequena descrição que ilustre o seu funcionamento. Assim, não caia na tentação de descrever as várias funções no relatório já que o resultado do `Doxygen` produzirá um documento com essa informação. Deverá configurar o `Doxygen` para gerar documentação para todas as entidades para garantir que não se esquece de nenhuma.

5.3 Critérios

Esta etapa será avaliada de acordo com os seguintes critérios:

1. Requisitos básicos (2 pontos dos 4 possíveis):
   (a) Deve existir um ficheiro chamado `makefile` que possibilite a utilização do comando `make` para compilar todo o projecto e gerar um executável chamado `letrorium`;
   (b) A `makefile` deve compilar os ficheiros com as opções `-Wall -Wextra -pedantic -ansi -O2`;
   (c) A compilação daí resultante não pode ter erros;
   (d) O programa deve ser capaz de ler os comandos a partir da linha de comandos;
   (e) O programa deve ser capaz de identificar correctamente os comandos e executá-los;
   (f) Os comandos devem funcionar todos correctamente em situações normais.
   (g) O relatório deve ser claro e documentar todas as opções de codificação tanto de estruturas de dados como da forma como os comandos foram lidos, interpretados e executados (não necessita de explicar função a função já que a documentação tem isso);
   (h) O código deve estar correctamente comentado e documentado (i.e., a documentação gerada pelo `Doxygen` deve ser completa, fidedigna e clara);
2. Requisitos médios (2 pontos dos 4 possíveis):

(a) O programa compilado com `-Wall -Wextra -pedantic -ansi -O2` não pode ter warnings;

(b) As estruturas de dados deve ser a mais eficiente possível tanto em termos de armazenamento como de facilidade de acesso;

(c) Devem reportar erros de inconsistência dos comandos (e.g., não se pode mudar a cor a uma célula se ainda não existe um puzzle);

(d) A implementação dos comandos deve ser eficiente;

(e) O código deve estar bem estruturado;

(f) As funções devem ser curtas[3] e usar funções auxiliares que implementem tarefas comuns;

(g) As linhas devem ser fácilmente compreensíveis: não deve ser necessário perder mais do que meio segundo por linha para perceber o que ela faz;

(h) Todas as funções devem ter nomes que ajudem a perceber o que elas fazem;

(i) Todas as variáveis devem ter nomes que ajudem a perceber a sua tarefa;

(j) Não devem existir variáveis globais.

[3] As linhas não devem ultrapassar os 60 caracteres e a função não deve ocupar mais do que um ecrã