Etapa 5 de Laboratórios de Informática II

Licenciatura de Engenharia Informática

Ano Lectivo 09/10

Versão 1.1

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**:

`imp`	Atribui cor à célula que tem maior impacto na resolução;
`rsv`	Resolve o puzzle;

`crg`	Corrige o puzzle;
`ger`	Gera um puzzle com um dado grau de dificuldade.

3 Célula de Maior Impacto

Considera-se que o impacto de uma célula é o número de células que são determinadas automáticamente pela determinação da sua cor. Assim, se ao atribuirmos a cor a uma célula estando `ab` e `ap` ligados isso faz com que mudemos a cor a 13 células então o impacto da célula é 13. Tendo em conta esta definição, pretende-se que o comando `imp` atribua a cor correcta à célula com maior impacto (de entre as que ainda não tem cor atribuída). No caso de empate, escolhe-se a que aparece primeiro (varrendo o tabuleiro linha a linha).

4 Resolução

A invocação do comando `rsv` resolve o puzzle. Isto é, atribui a cor correcta a todas as células que faltam no tabuleiro.

5 Correcção

A invocação do comando `crg` corrige o puzzle. A razão de ser deste comando é ser invocado quando sabemos que o puzzle não tem resolução e não queremos recomeçar mas simplesmente andar para trás na nossa resolução até um ponto em que o puzzle ainda tenha solução.

6 Geração de Puzzles

Esta tarefa é só para os alunos que quiserem ter mais de 17 à disciplina.

Usando o comando `ger` seguido de um número inteiro e de uma palavra (`facil`, `medio` ou `dificil`) o seu programa deverá gerar um puzzle válido desse tamanho. Será avaliado mais favoravelmente se for possível indicar dificuldades diferentes (e.g., fácil, média, difícil). Este comando só será avaliado para grupos cuja classificação seja de pelo menos 17/20.

Uma nota importante: o gerador só deve gerar problemas com uma única solução.

7 Avaliação

Esta etapa vale no máximo 6 valores e deve ser entregue até ao fim do dia 30 de Maio no site da disciplina. Para não reprovar à disciplina os alunos precisam de tirar um mínimo de 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.

7.1 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 quinta etapa pelo grupo número 3 do turno PL1 teria como nome `g3PL1-et5.tar.bz2`.

7.2 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.

7.3 Critérios

Esta etapa será avaliada de acordo com os seguintes critérios:

1. Requisitos básicos (2 pontos dos 6 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 ponto dos 6 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[1] 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. Requisitos avançados (2 pontos dos 6 possíveis):

   (a) Gerar puzzles com diferentes dificuldades.

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