

Etapa 2 de Laboratórios de Informática II

Licenciatura de Engenharia Informática

Ano Lectivo 09/10

Versão 1.1

Conteúdo

1 Critérios de Aprendizagem	1
2 Enunciado	2
3 Pares desencontrados	2
4 Avaliação	3
4.1 Material a entregar para cada uma das etapas	3
4.2 Relatório	3
4.3 Critérios	4

1 Critérios de Aprendizagem

No final desta etapa os alunos deverão ser capazes de:

1. Compilar programas em código C utilizando o gcc e correr o executável gerado pelo compilador;
2. Utilizar intruções condicionais, ciclos e funções;
3. Armazenar e percorrer matrizes;
4. Utilizar a biblioteca de input/output do C para ler e escrever ficheiros de texto;
5. Fazer algum processamento básico sobre *strings*;
6. Percorrer matrizes e efectuar alguns algoritmos sobre estas estruturas de dados.

2 Enunciado

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

trp	Procura triplos no tabuleiro e atribui correctamente as cores que consegue inferir;
snd	Procura sandes no tabuleiro e atribui correctamente as cores que consegue inferir;
pis	Procura pares isolados no tabuleiro e atribui correctamente as cores que consegue inferir;
pds	Procura pares desconstrados no tabuleiro e atribui correctamente as cores que consegue inferir;
vb	Implica percorrer todo o tabuleiro e ver se numa mesma linha ou coluna não há duas letras iguais com a cor branca;
vp	Implica percorrer todo o tabuleiro e ver se não existem duas células pretas vizinhas ortogonalmente;
vl	Implica percorrer todo o tabuleiro e ver se não existem duas células brancas ou indiferentes que não tenham um caminho entre elas (sem passar por células pretas).

3 Pares desconstrados

Esta estratégia utiliza-se quando numa linha (ou coluna) temos duas letras A e B e nas mesmas colunas (ou linhas) temos as mesmas letras que se tocam na diagonal. Nesse caso, as letras da outra diagonal do mesmo quadrado são brancas.

Segue-se um exemplo. Neste exemplo, os caracteres X representam letras (não necessariamente o X) que não interessam para explicar a situação em causa. Neste caso, as letras C e D tem que ser brancas. Tente perceber porquê.

X	X	X	X	X	X	X	X
X	X	A	B	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	A	D	X	X	X	X
X	X	C	B	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X

4 Avaliação

Esta etapa vale no máximo 3 valores e deve ser entregue até ao fim do dia 3 de Abril no site da disciplina. Para não reprovar à disciplina os alunos precisam de tirar um mínimo de 50% da classificação desta etapa (1.5 valores). A entrega atrasada incorre numa penalização de 10% ao dia.

A avaliação é feita durante as aulas da semana seguinte. Caso um aluno não apareça à avaliação sem apresentar motivos de força maior devidamente justificados por documentos este reprova automaticamente à disciplina por faltar à avaliação.

4.1 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 <code>makefile</code> utilizada para compilar todo o código e gerar a documentação;
doc	Uma pasta com a documentação em formato <code>html</code> gerada a partir do código utilizando a ferramenta <code>Doxygen</code> ;
relatorio.pdf	Um ficheiro gerado utilizando o comando <code>L^AT_EX</code> no formato <code>pdf</code> com o relatório;
doc.pdf	Um ficheiro gerado utilizando o comando <code>L^AT_EX</code> a partir dos ficheiros gerados pela ferramenta <code>Doxygen</code> .

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 segunda etapa pelo grupo número 3 do turno PL1 teria como nome `g3PL1-et2.tar.bz2`.

4.2 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 automaticamente 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.

4.3 Critérios

Esta etapa será avaliada de acordo com os seguintes critérios:

1. Requisitos básicos (1.5 ponto dos 3 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 a opção **-Wall**;
 - (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 (1.5 ponto dos 3 possíveis):
- (a) O programa compilado com -Wall não pode ter warnings;
 - (b) As estruturas de dados deve ser a mais eficiente possível tanto em termos de storage 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 e usar funções auxiliares que implementem tarefas comuns;
 - (g) Todas as funções devem ter nomes que ajudem a perceber o que elas fazem;
 - (h) Todas as variáveis devem ter nomes que ajudem a perceber a sua tarefa;
 - (i) Só pode existir **no máximo** uma variável global.