Projecto de Laboratórios de Informática II

Illuminatus

2011/2012

O projecto deste ano consiste em implementar na linguagem de programação `C` no sistema operativo `Linux` (disponibilizado na máquina virtual) uma aplicação para resolver um tipo de puzzle chamado Illuminatus. O objectivo deste puzzle é iluminar todas as casas desbloqueadas dum quadriculado.

Uma casa é iluminada se:

- Tiver uma lâmpada;
- Tiver uma lâmpada na mesma linha ou coluna e não existir nenhuma casa bloqueada entre a casa e a que contém a lâmpada.

Quando uma casa está bloqueada esta pode conter um número. Neste caso, o número obriga a que existam esse número de lâmpadas nas casas livres adjacentes (só contam as casas ortogonais e não as diagonais).

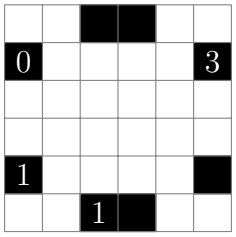
Existe uma restrição para a colocação de lâmpadas: uma lâmpada não pode iluminar outra lâmpada, isto é, não podem existir duas lâmpadas na mesma linha ou coluna a não ser que entre elas esteja pelo menos uma casa bloqueada.

Há várias estratégias para a resolução destes puzzles:

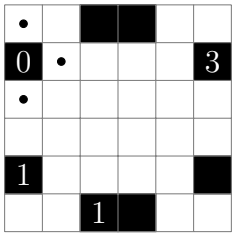
1. Se uma casa com um número n tem exactamente n casas vizinhas onde possam existir lâmpadas então todas essas casas vizinhas tem que conter lâmpadas;
2. Se uma casa contém uma lâmpada pode-se sombrear todas as casas na mesma linha e coluna que não estejam bloqueadas;
3. Podem-se colocar marcas numa casa quando se têm a certeza que esta casa não pode conter uma lâmpada; isto pode acontecer nas casas vizinhas dum 0 ou em casas onde colocar uma lâmpada criaria uma contradição (por exemplo não se pode colocar uma lâmpada numa casa que seja vizinha diagonalmente de uma casa com um 4);
4. Nos casos em que hajam dois números numa diagonal, por exemplo 1 e 3 temos a certeza que as duas casas partilhadas não podem ser as duas iluminadas ao mesmo tempo; logo as outras duas casas têm que conter ambas lâmpadas;
5. Uma técnica mais avançada consiste em verificar se uma casa não iluminada só pode ser iluminada se uma dada casa tiver uma lâmpada (devido a outras restrições).

Exemplo

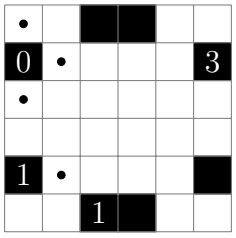
Segue-se um exemplo de resolução. Veja o quadro seguinte e tente encontrar uma solução.



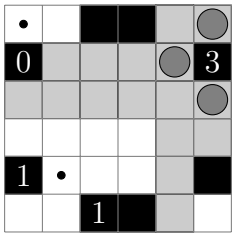
Repare que podemos marcar as três casas vizinhas do 0 porque não podem conter lâmpadas.



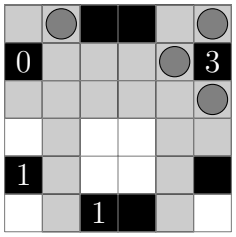
Repare a casa que se assinalou agora como não podendo conter uma lâmpada. Se essa casa tivesse uma lâmpada não era possível fazer o 1 da linha de baixo já que nenhuma das casas vizinhas desse 1 podia conter uma lâmpada.



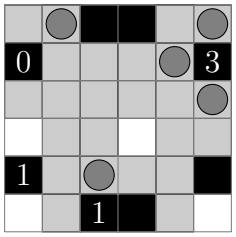
Podemos agora colocar as lâmpadas à volta do 3 e iluminar as casas nas mesmas linhas e colunas que essas lâmpadas.



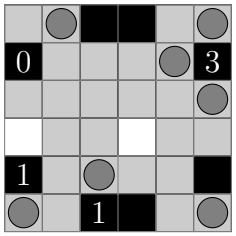
Só existe uma maneira de iluminar o canto superior esquerdo: colocar uma lâmpada à sua direita.



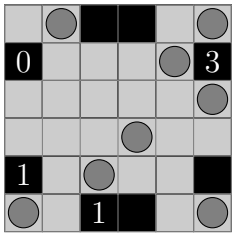
Agora temos mais uma aplicação da regra básica que nos permite colocar mais uma lâmpada.



Ficamos agora com duas casas que só podem ser iluminadas se colocarmos uma lâmpada nessa casa.



Resta-nos uma hipótese já que uma casa com um 1 só pode ter uma lâmpada vizinha.



Calendarização

Etapa	Data de Entrega
1ª etapa	25-03-2012
2ª etapa	13-05-2012
3ª etapa	3-06-2012

Tarefas

As tarefas são classificadas de A a E (sendo o A a dificuldade mínima e o E a mais avançada). Cada tarefa vale 1 ponto se for efectuada na etapa planeada e menos se for acabada em etapas seguintes de acordo com a tabela seguinte:

Etapa planeada	Feita na 1ª etapa	Feita na 2ª etapa	Feita na 3ª etapa
1	1	0.8	0.5
2	1	1	0.8
3	1	1	1

Para obter aprovação à disciplina é necessário obter a classificação de 10 valores. Segue-se a lista das tarefas:

Etapa 1

1. $_{(A)}$. Interpretador de comandos;
2. $_{(A)}$. Ler o estado do puzzle de ficheiro e escrever o estado do puzzle em ficheiro;
3. $_{(A)}$. Colocar/remover uma lâmpada numa casa;
4. $_{(A)}$. Implementar a estratégia 1;
5. $_{(A)}$. Implementar a estratégia 2;

Etapa 2

6. $_{(B)}$. Implementar a estratégia 3;
7. $_{(B)}$. Implementar a estratégia 4;
8. $_{(C)}$. Implementar a estratégia 5;
9. $_{(A)}$. Fazer um módulo de listas ligadas;
10. $_{(B)}$. Fazer um sistema de anulação de comandos;

Etapa 3

11. $_{(D)}$. Resolver um puzzle;
12. $_{(A)}$. Analisar o código *assembly* gerado pelo compilador a partir de uma função (em C) e criar a tabela de alocação de registos;
13. $_{(C)}$. Optimização do código e análise da complexidade do código usando as ferramentas `gprof` e `gcov`;
14. $_{(A)}$. Compilar o código no sistema operativo `Linux` sem avisos (*warnings*) nem erros com as opções `cc -ansi -Wall -Wextra -pedantic -O4 -lreadline`;
15. $_{(E)}$. Gerar um puzzle com uma única solução;
16. $_{(C)}$. Interface gráfico.
17. $_{(A)}$. Documentação do código (Doxygen);
18. $_{(C)}$. Legibilidade do código e tratamento de erros;
19. $_{(A)}$. Assiduidade e participação;
20. $_{(A)}$. Gestão de Projecto;

Gestão de Projeto

Os grupos de trabalho são compostos por 3 elementos e terão necessáriamente de ser compostos por pessoas do mesmo turno prático. Nos casos em que o número de elementos no turno não seja divisível por 3 aceitam-se 2 grupos de 2 elementos se o resto da divisão do número por 3 for 1 e 1 grupo de 2 elementos se o resto der 2.

Pretende-se que os grupos de trabalho façam a gestão do projecto, utilizando para esse efeito o `Redmine`, evidenciando desse modo a sua capacidade de organização. Para o efeito será utilizada uma ferramenta de gestão de projectos que permitirá ao grupo planear o desenvolvimento do projecto, definindo subtarefas, fazer a sua atribuição aos elementos da equipa e acompanhar a sua implementação. Deverão também utilizar todas as restantes funcionalidades da ferramenta (e.g. documentação, wiki, etc.). A utilização correcta do sistema de gestão de projeto é obrigatório para a avaliação.

Não serão avaliadas as tarefas que não tenham sido correctamente introduzidas e contabilizadas no sistema.

Material a entregar em cada etapa

- Documentação gerada automáticamente pelo `Doxygen`;
- Relatório de desempenho do grupo na execução das diversas tarefas utilizando funcionalidades da ferramenta de gestão de projecto (descrição das tarefas incluindo tempo total dispendido e tempo por cada pessoa envolvida);
- Código fonte e respectiva `makefile`.

Critérios obrigatórios

Os seguintes critérios tem que ser cumpridos ou a entrega não é válida:

- O programa tem compilar sem erros e funcionar na máquina virtual disponibilizada;
- A makefile tem que compilar com as opções `-Wall -Wextra -pedantic -ansi -O4`;
- O nome do executável tem que ser `illuminatus`;
- O programa tem que ler os comandos do `stdin`.

Entrega

Deverá ser colocado na opção "Files"do redmine ("Ficheiros"para os alunos que usam a versão portuguesa) um arquivo compactado com o comando `tar` do qual constem os seguintes ficheiros e pastas:

identificacao ficheiro com a identificação dos alunos (nome completo e número);

code pasta com o código fonte e respectiva `makefile`,

doc com a documentação `html`[1] gerada pelo `doxygen`.

A pasta deverá ter o nome PL<nº do turno>g<nº do grupo>-et<nº da etapa>.tar.bz2
Nos casos em que o número do grupo seja só um algarismo este deverá ser precedido de um zero. Exemplos:

- PL6g02-et1.tar.bz2 grupo 2 do turno 6 a entregar a etapa 1
- PL2g11-et2.tar.bz2 grupo 11 do turno 2 a entregar a etapa 2

Para se usar o comando `tar` da forma correcta

1. Abre-se uma consola no Linux
2. Usando o comando cd vai-se para a directoria que contém as pastas code e doc
3. Escreve-se o comando tar jcf PL2g11-et2.tar.bz2 identificacao code doc

Se não cumprirem alguns destes requisitos, o trabalho não será considerado entregue.

[1] i.e., deve conter dentro (e não em subpastas) o ficheiro index.htm ou index.html gerado pelo Doxygen assim como os restants ficheiros nnecessários