Universidade do Minho
Licenciatura em Engenharia Informática
Laboratórios de Informática II

Etapa II

Ano Lectivo de 2009/2010

54738 **João Gomes**
54745 **André Pimenta**
54764 **Nelson Carvalho**

3 de Abril de 2010

Resumo

Neste relatório está exposta a realização da segunda etapa do Projecto da Unidade Curricular de **Laboratórios de Informática II**, integrada no plano de estudos do primeiro ano da LEI.

Esta etapa consistui no desenvolvimento de um módulo de ajuda do jogo **"Letrorium"**, e pode ser vista como um melhoramento do trabalho desenvolvido na primeira etapa.

Conteúdo

Capítulo 1

Introdução

1.1 Contextualização e apresentação do Caso de Estudo

O presente relatório tem como objectivo explicar os passos dados pelo grupo na realização da segunda etapa do projecto da Unidade Curricular de **Laboratórios de Informática II**, que, como já foi dito, consistiu em desenvolver um módulo de ajuda que, utilizando um determinado comando, procurava por erros no jogo **"Letrorium"**.
Esta etapa exigiu uma aplicação acrescida em relação á primeira, e as dificuldades que o grupo encontrou no seu desenvolvimento, assim como a forma em como as ultrapassou, serão aqui descritas.

1.2 Motivação e objectivos

À partida para a resolução da etapa, esta carregava um pouco mais de motivação, em relação à primeira: os conhecimentos da linguagem C que exigia eram mais aprofundados, e, por ser um trabalho que exigia não só capacidades de programação na referida linguagem, mas também uma boa capacidade de raciocínio, resolvê-la tornou-se um problema interessante.
Esta etapa tem como principal objectivo fazer com que os alunos aprofundem os conhecimentos da linguagem C adquiridos durante a realização da primeira etapa, pois nela foram testadas as nossas capacidades para manipulação de informação em matrizes e a utilização de ciclos, funções ou instruções condicionais, por exemplo.

1.3 Estrutura

O presente relatório é constitúido por três capítulos: **Introdução**; **Desenvolvimento**, divido em duas secções, **Análise do Problema**, **Solução do Problema** e **Ferramentas utilizadas**, onde explicamos mais detalhadamente os passos dados pelo grupo durante a realização da etapa, assim como as dificuldades que foram surgindo e a forma em como as ultrapassámos; e, por fim, a **Conclusão**.

Capítulo 2

Desenvolvimento

2.1 Análise do Problema

Como já foi dito, esta etapa exigia que fossem desenvolvidos uma série de comandos que servissem de ajuda ao utilizador do "**Letrorium**". Esses comandos podem ser considerados um melhoramento a acrescentar aos já desenvolvidos na primeira etapa.
Os comandos a implementar no **"Letrorium"** eram os seguintes:

trp, que permite ao utilizador procurar triplos no trabuleiro, atribuindo depois correctamente, ás células em questão, as cores que conseguir inferir;

snd, que procura "sandes" no tabuleiro (e.g. ABA), atribuindo ás células em questão as cores correctas;

pis, que, procurando pares "isolados" no tabuleiro, atribui ás células em questão as cores correctas;

pds, que permite ao utilizador procurar pares "desencontrados" no tabuleiro, atribuindo depois ás células em questão as cores correctas;

vb, que, percorrendo todo o tabuleiro, verifica se não existem numa mesma linha ou coluna duas células iguais de cor branca;

vp, que percorre todo o tabuleiro e verifica se não existem duas células de cor preta, vizinhas ortogonalmente;

vl, que percorre todo o tabuleiro e verifica se não existem duas células brancas ou de cor indefinida sem um caminho entre elas (que não passem por células pretas).

Os comandos **vb**, **vp** e **vl** são funções de verificação que reportarão erros nas jogadas.

Alguns destes comandos, assim como a forma em como foram implementados, serão explicados mais detalhadamente na próxima secção, **"Resolução do Problema"**.

2.2 Solução do Problema

Descreveremos agora os comandos de ajuda desenvolvidos que achamos mais importantes e as dificuldades que encontrámos ao longo da resolução da etapa.
Para começar, é importante referir que o trabalho desenvolvido na primeira etapa foi de grande importância para a resolução desta. A função **executa_comando**, definida na primeira fase, foi utilizada para seleccionar e executar os comandos desenvolvidos nesta. Depois, a estrutura de dados que usámos logo no início do projecto facilitou de sobremaneira o trabalho do grupo nesta etapa.

A estrutura de dados utilizada na primeira fase foi a seguinte:

```
 typedef struct elemento{
    char letra;
    char cor;
} Elemento,*Elem;
```

Vamos agora explicitar os comandos de ajuda/ verificação desenvolvidos nesta etapa. Explicaremos a sua finalidade e a forma em como os construímos.

trp - Triplos

O comando **trp** permite ao utilizador do **"Letrorium"** procurar por triplos no tabuleiro. Como a própria palavra indica, este comando deve indicar se existem na matriz três letras iguais, sequenciais.
Para definirmos esta função, utilizámos dois ciclos que nos ajudam a percorrer o tabuleiro. Depois de colocados numa qualquer célula, verificamos se a célula imediatamente antes e a imediatamente a seguir são iguais áquela em que nos encontramos. Se isso acontecer e se a as células anterior e a seguir forem de cor branca e a célula em que nos encontramos for de cor preta, a função deve devolver um erro (**E_WRONG_SOLUTION**). Caso contrário, função "pintará" as três células de preta, branca e preta, por esta ordem.

snd - Sandes

O comando **snd** serve para procurar "sandes" no tabuleiro (ou seja, procurar por três células sequenciais que tenham nos extremos a mesma letra, e uma letra diferente na célula central,e.g. ABA). Para definir esta função utilizámos um algoritmo parecido com o utilizado no comando **trp**, sendo que, desta vez, a função devolve um erro apenas se a célula central for preta. Caso isso não aconteça, a função "pinta" a referida célula de cor branca.

pis - Pares isolados

O comando **pis** verifica se, caso numa linha/coluna existe um par de letras, e, nessa mesma linha/coluna existe uma letra "isolada", a célula da letra em questão deve ser pintada de preta. A função devolve o erro **E_WRONG_SOLUTION** caso a cor da célula "isolada" seja branca. Para desenvolvermos este comando, contámos novamente com a ajuda de dois ciclos **for** para percorrer o tabuleiro. De seguida, armazenámos a letra da referida célula numa variável, para confirmar se esta é mesmo "isolada", ou seja, se na sua vizinhança não existe nenhuma célula com a mesma letra.

pds - Pares desencontrados

O conceito de "par desencontrado" é simples: se numa mesma linha/ coluna existe um par de células de letra diferente (e.g. AB), na linha/ coluna onde se encontrem "A" e "B" não podem existir duas outras células com as letras "A" e "B" cujas diagonais se toquem. No entanto, o comando **pds** foi um dos que mais dificuldades nos apresentou no desenvolvimento de toda a etapa. Conseguimos ainda assim resolver os problemas que este comando nos estava a dar utilizando um algoritmo algo parecido com o utilizado em **pis**.
Assim, em vez de guardarmos o valor de uma só célula, guardamos o de um "quadrado", ou seja, de quatro células, verificando de seguida a diagonal do quadrado onde se encontrasse o "par desencontrado". Se as duas células dessa diagonal forem pretas, o programa deve devolver **E_WRONG_SOLUTION**. Caso contrário, a função deve transformar a referida diagonal em cor branca.

Esta etapa exigia também que se implementassem também comandos de verificação de erros no tabuleiro. Estes comandos, depois de desenvolvidos, devem apenas reportar alguns erros qe o tabuleiro do **"Letrorium"** tenha, não alterando o seu estando.
Os comandos de verificação que desenvolvemos são os seguintes:

vb

O comando **vb** deve percorrer todo o tabuleiro, procurando por duas céluas com a mesma letra, de cor branca, na mesma linha/coluna.
Para procurar pelas referidas células, foram necessários três ciclos **for**, dois para percorrer as células do tabuleiro, uma a uma, e um terceiro para comparar a célula em que nos encontremos com todas as da sua linha/ coluna.

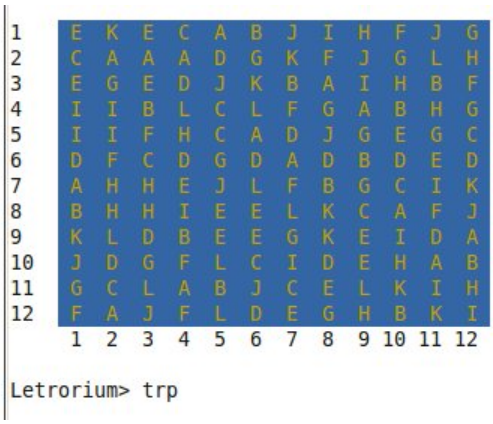
vp

Como já foi dito, o comando **vp** deve percorrer todo o tabuleiro e verificar se não existem duas células de cor preta, vizinhas ortogonalmente. O algoritmo que criámos para o desenvolver é também algo simples: novamente com a ajuda de dois ciclos **for**, percorremos as células da matriz uma a uma. Se uma das células imediatamente a baixo ou imediatamente á direita da célula em que nos encontramos for também de cor preta, a função devolve o erro **E_WRONG_SOLUTION**.

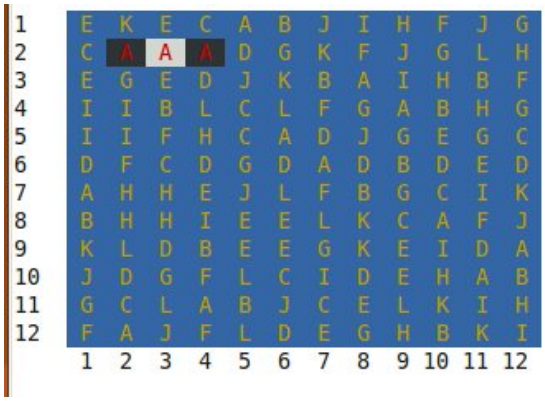
vl

O comando **vl** foi, a par do comando **pds**, o mais complexo de todos os implementados nesta etapa. Este deve verificar se, percorrendo todo o tabuleiro, não existem duas células brancas ou de cor indefinida sem um caminho entre elas, ou seja, sem necessidade de "atravessar" células de cor preta.
Para implementarmos esta função, criámos uma matriz auxiliar, com a mesma dimensão e de acordo com as informações do nosso tabuleiro: onde, na nossa matriz se encontrar uma célula branca ou indefinida, na mesma posião da matriz auxiliar será guardado um 1; para as células pretas, será feito o mesmo, mas guardando um 0.
Seguindo este raciocínio a matriz auxiliar sofre mais algumas alterações, explicitadas no nosso código, que nos dizem se existe ou não caminho entre quaisquer duas células brancas ou de cor indefinida.
É este comando que determina se a solução geral do tabuleiro está ou não correcta.

Exibiremos agora duas imagens que ajudam a explicar um dos comandos que desenvolvemos nesta etapa, o **trp**, que procura por triplos no tabuleiro.



Inserindo o comando **trp**, ele é executado e o seu resultado é o seguinte:



Como podem confirmar pela imagem, as células que compõem o triplo encontrado no tabuleiro são pintadas de preto, branco e preto, respectivamente.

2.3 Ferramentas utilizadas

Durante esta etapa, as ferramentas mais utilizadas pelo grupo foram as seguintes:

Kile - Editor gráfico para LATEX;

NetBeans - IDE de código aberto para desenvoler software nas linguagens Java, C/ C++ , PHP e outras;

gedit - editor oficial de texto plano para o Gnome.

É importante referir que todos os elementos do grupo utilizaram neste projecto a distribuição do Sistema Operativo Linux, **Ubuntu**.

Capítulo 3

Conclusão

Depois de terminada a segunda etapa do projecto desta Unidade Curricular, concluimos que os seus objectivos iniciais foram atingidos com êxito.
Como já foi dito anteriormente no relatório, esta etapa carregava consigo um maior grau de motivação, por se tratar de um trabalho mais interessante, mas também um maior grau de dificuldade. Desenvolver comandos como o **pds** ou **vl** revelou-se uma tarefa complicada, no entanto, sentimos que ultrapassámos com sucesso os problemas que foram surgindo. Prova disso é o facto de, no script que os docentes responsáveis desenvolveram para avaliar esta etapa, o nosso trabalho passar em todos os 27 testes.
Assim, o grupo julga que o trabalho desenvolvido foi razoavelmente satisfatório.