Universidade do Minho
Licenciatura em Engenharia Informática
Laboratórios de Informática II

Etapa III

Ano Lectivo de 2009/2010

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

2 de Maio de 2010

Resumo

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

Esta etapa consistiu principalmente no desenvolvimento de comandos automáticos de atribuição de cor e da anulação de jogadas, no tabuleiro do jogo **"Letrorium"**.

Conteúdo

Conteúdo **1**

1 Introdução **2**
1.1 Contextualização e apresentação do Caso de Estudo 2
1.2 Motivação e objectivos 2
1.3 Estrutura 2

2 Desenvolvimento **3**
2.1 Análise do Problema 3
2.2 Solução do Problema 4
2.3 Ferramentas utilizadas 8

3 Conclusão **9**

Capítulo 1

Introdução

1.1 Contextualização e apresentação do Caso de Estudo

O presente relatório serve para explicar detalhadamente os passos dados pelo grupo de trabalho ao longo da realização da terceira etapa do projecto da Unidade Curricular de **Laboratórios de Informática II**, do primeiro ano da Licenciatura em Engenharia Informática.
O trabalho desta fase consistiu essencialmente na elaboração de comandos automáticos de atribuição de cores e anulação de jogadas/ movimentos, tarefas estas referentes ao jogo de tabuleiro que tem vindo a ser desenvolvido nesta UC, de nome **"Letrorium"**.

1.2 Motivação e objectivos

A motivação do grupo, à partida para esta etapa, estava num patamar elevado. O nível de dificuldade desta etapa aumentou, mas o interesse que a mesma provoca nos alunos também, uma vez que passou a exigir conhecimentos sobre como se trabalha com memória na linguagem de programação C. O facto de esta etapa estar cotada, em termos de avaliação, para quatro valores, em vez dos três valores das habituais, fez também com que a motivação do grupo fosse algo superior.
Esta etapa tem como principais objectivos tornar os alunos capazes de efectuar gestão de memória (alocação e libertação de espaço), utilizar listas ligadas e utilizar depuradores de código (**GDB e Valgrind,** p.e.), que ajudam os utilizadores a encontrar/ corrigir erros no código desenvolvido.

1.3 Estrutura

O presente relatório é constitúido por três capítulos: **Introdução**; **Desenvolvimento**, dividido 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 dissemos, esta etapa consistiu no desenvolvimento de comandos que permitam ao utilizador do jogo fazer atribuição automática de cores ás células do tabuleiro e, também, de comandos de anulação de jogadas/ movimentos.
Os comandos a implementar no **"Letrorium"** eram os seguintes:

ab, que liga/ desliga o automático branco. Este comando deve "pintar" de preto todas as células da mesma linha e coluna que uma célula que acabou de ser pintada de branco;

ap, com um funcionamento parecido com o do comando **ab**, mas que, neste caso, deve pintar de branco apenas as quatro células vizinhas ortogonalmente da célula que acabou de ser pintada de preto;

anc, que anula a cor. Este comando deve anular a última atribuição de cor efectuada;

anm, que permite ao utilizador anular o último movimento efectuado;

it, que inicia uma tentativa. Este comando cria uma marca que permite ao utilizador do jogo saber o estado do tabuleiro para onde passar quando for utilizado o comando **rb**;

rb, comando "Roll Back". Faz-nos voltar ao estado do tabuleiro aquando da última invocação do comando **it**;

st, que verifica o estado do tabuleiro. Deve imprimir a mensagem **E_WRONG_SOLUTION**, no **STDERR** (utilizado **mensagem_de_erro**) caso este não seja válido.

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 de forma mais específica os comandos desenvolvidos nesta terceira etapa. Comecemos por descrever as três estruturas de dados criadas propositadamente para esta etapa e que foram extremamente importantes nesta fase do projecto, facilitando sobremaneira a tarefa do grupo.

Em primeiro lugar, a estrutura a que chamámos de **jogadas**:

```
typedef struct jogadas{
int col;
int row;
char color;
struct jogadas* next;
}Jogadas;
```

Esta estrutura permitiu-nos guardar nas variáveis **col**, **row** e **color** a coluna, a linha e a cor, respectivamente, dos valores que se encontravam nessa célula antes de ser efectuada a jogada, e foi importante no desenvolvimento dos comandos **anc**, **anm**, **it** e **rb**.

De seguida, explicaremos a estrutura **movimentos**:

```
typedef struct movimentos{
int pos;
struct movimentos *next;
}Movimentos;
```

Esta estrutura guarda na variável **pos** o número de cores que foram gravadas até ao momento, e os comandos onde a sua utilização foi importante são os referidos na descrição da estrutura **jogadas**.

Por último, descreveremos a estrutura **gravados**:

```
typedef struct gravados{
int it;
int totalGr;
Jogadas *jgs;
Movimentos *mvs;
}Gravados;
```

Esta estrutura contém os dados das estruturas **jogadas** e **movimentos** e ajuda-nos a aceder às jogadas e aos movimentos efectuados.

Nesta parte do relatório, vão ser explicados de forma mais específica os comandos desenvolvidos pelo grupo nesta etapa. As dificuldades que encontrámos no seu desenvolvimento serão também descritas, assim como as ideias que tivemos para as ultrapassar.

Começeremos por explicar os comandos de atribuição de cor automáticos que desenvolvemos, **ab** e **ap**.

O comando **ab**, como já foi dito, liga o "automático branco". A sua função é, quando activo, pintar de preto todas as células de mesma linha/ coluna de uma célula que tenha acabado de ser pintada dessa cor, se contiverem também a mesma letra dessa célula.
Para se saber quando é que este comando está ou não activo, criámos na função **main**, definida no início deste projecto, uma variável tipo *flag* (flagAB) que tem o valor de 0 quando o comando está desligado e de 1 quando está ligado.
Para que o "automático branco" seja novamente desactivado, deve ser introduzido a qualquer altura novamente o comando **ab**.

O comando **ap** tem um funcionamento parecido ao **ab**. A diferença entre este e o anteriormente explicado é que deve pintar de branco não os elementos que se encontrem na mesma linha/ coluna da célula pintada desta de preto, mas sim as células que sejam vizinhas desta ortogonalmente.
A forma em como o comando **ap** foi idealizado e desenvolvido é, no resto, igual á do comando **ab**.

Serão agora descritos os comandos de anulação de cor e de movimento, respectivamente **anc** e **anm**.

O comando **anc** deve eliminar a última atribuição (ou as últimas atribuições, caso mais do que uma célula sofra alterações) de cor efectuada.
Para definirmos este comando, criámos uma lista ligada onde as cores do tabuleiro vão sendo guardadas, conforme as alterações que vão sendo feitas. Quando o comando **anc** é executado, o tabuleiro volta a ter as cores da jogada anterior, sendo as actuais eliminadas também da lista ligada.

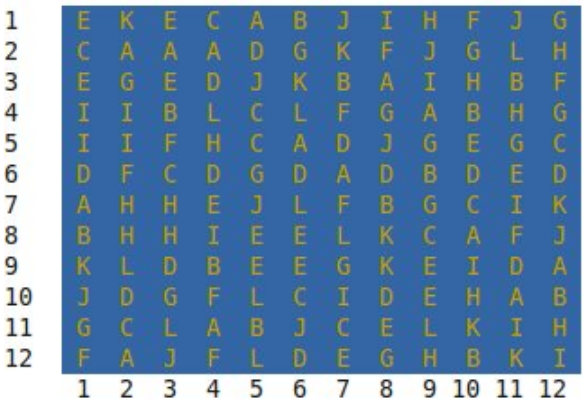
No que diz respeito ao comando **anm**, este vai anulando cores atribuídas correspondentes ao último movimento, sendo apenas interrompido quando o estado anterior do tabuleiro for restabelecido.

Por fim, descreveremos os comandos **it** e **rb**, directamente ligados.

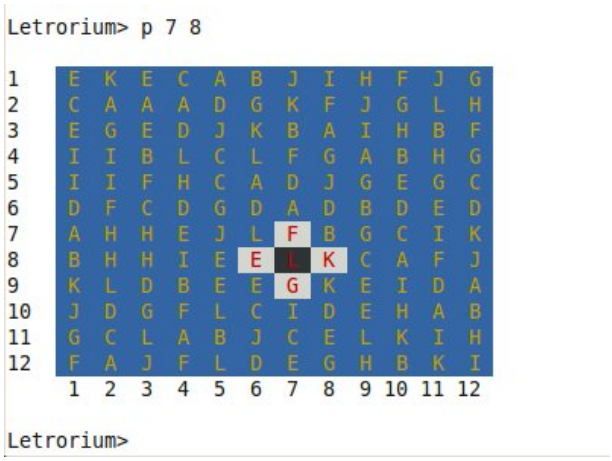
O comando **it** cria uma marca do tabuleiro, guardando o seu estado, estado esse que pode ser retomado quando o utilizador quiser, recorrendo ao comando **rb**.
Para isso, foi necessário, a nível de código, acrescentar à lista ligada de its a posição a que corresponde o tabuleiro actual, isto na estrutura de jogadas gravadas.

O comando **rb**, ou "roll back", quando accionado, restabelece o estado do tabuleiro guardado aquando da última vez que fora accionado o comando **it**.
Para o desenvolvermos, foi necessário ir anulando todas as atribuições de cores efectuadas até chegar ao estado do tabuleiro marcado através da última introdução do comando **it**.

Exibiremos agora duas imagens que ajudam a explicar um dos comandos que desenvolvemos nesta etapa, **ap**.



Primeiro, é inserido o comando em questão, **ap**.



Depois, é inserido o comando **p 7 8** e as células vizinhas ortogonalmente à célula posicionada na linha 7 e na coluna 8 são, de forma automática, pintadas de branco.

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

Concluída a terceira etapa do projecto, o grupo pensa ter atingido todos os objectivos propostos pelos professores responsáveis no início da mesma.
O nível de exigência desta fase aumentou consideravelmente em relação às duas anteriores, uma vez que havia a necessidade de manipular listas ligadas e fazer uma boa gestão de memória para fazer com êxito das tarefas propostas. No entanto, esse nível de exigência superior em termos de conhecimento da linguagem C não fez com que o grupo de trabalho não conseguisse ultrapassar, com menor ou maior dificuldade, os problemas que foram surgindo no decorrer da etapa. Uma das provas disso é o facto de o código desenvolvido passar com êxito em todos os 141 testes do script de avaliação que os professoeres responsáveis desenvolveram.
Visto isto, o grupo de trabalho fica agora expectante em relação à próxima fase do trabalho.