Lab 2

Relatório Final

Mestrado Integrado em Engenharia Informática e Computação

Redes de Computadores

Grupo 3:
Francisco Rodrigues - 201305627
João Nogueira - up201303882
Marta Lopes - 201208067

Faculdade de Engenharia da Universidade do Porto
Rua Roberto Frias, sn, 4200-465 Porto, Portugal

20 de Dezembro de 2015

1 Introdução

Este relatório tem como objetivo explicar o segundo projeto da Unidade Curricular de Redes de Computadores bem como analisar os resultados obtidos na realização das experiências especificadas no enunciado do mesmo.

Este projeto encontra-se dividido em duas grandes partes. Em primeiro lugar, é-nos pedido que desenvolvamos uma aplicação de *download* que proceda à transferência de um ficheiro e que implemente o protocolo *FTP*. Em segundo lugar, é-nos pedido que configuremos e estudemos uma Rede de Computadores seguindo a estrutura das experiências abaixo enumeradas:

1. Configuração de um *IP* de rede;
2. Configuração de duas Redes *LAN* virtuais num *switch*;
3. Configuração de um *router* em *Linux*;
4. Configuração de um *router* comercial implementando *NAT*;
5. *DNS*;
6. Conexões *TCP*.

Conteúdo

2 Parte 1 - Aplicação de download

Como referido anteriormente, a primeira parte deste tranalho consiste numa aplicação que transfere um ficheiro utilizando o protocolo *FTP* descrito no ficheiro RFC959. Como método de *input* é utilizada a sintaxe mostrada na figura abaixo como descrito no ficheiro RFC1738.

```
joao@joao-VirtualBox:~/Documents/Github/RCOM-code/ftpDownloader$ ./download ftp://
PROGRAM: Entering anonymous Mode
ERROR - Wrong paramater -> URL | Expected something like: ftp://[<user>:<password>@]<host>/<url-path>
```

Figura 1: Input

A aplicação desenvolvida permite que seja feito um download em modo anónimo. Para tal basta não colocar os caracteres '@' e ':' e não colocar nome de utilizador e password. Neste caso a aplicação irá assumir o utilizador *anonymous* e a palavra-passe vazia.

2.1 Arquitetura

A *UrlStruct* é a estrutura definida respponsável por guardar a informação necessária que depende do *input* do utilizador.

```
1 typedef struct{
2     char * user;
3     char * password;
4     struct hostent * h;
5     char * urlPath;
6     char * hostIp;
7 }urlStruct;
```

urlStruct

Ao correr o programa é chamada a função *getUrlInfo* que é responsável por pegar na *string* que o utilizador forneceu como argumento e interpretar toda a informação necessária.

```
1 #include "utilities.h"
2
3 void getUrlInfo(char * completeString, urlStruct * url);
```

Url Header

3 O Jogo - Q!nto

3.1 História

"Q!nto" é um jogo de tabuleiro recente, criado no ano de 2014 no qual podem participar entre 2 e 4 jogadores. Foi criado por Gene Mackles e lançado pela PDG games

Este jogo faz parte de uma coleção de três jogos chamada Triple Play que é uma coleção de jogos de tabuleiro.

3.2 Conteúdo do Jogo

Estão incluídas no Jogo 60 cartas divididas como mostra a figura à direita.

3.3 Regras de Jogo

Existem três versões para jogar o Q!into, sendo elas o *Classic*, *Plus* e *Light*. Neste projeto usaremos apenas a versão Light que passamos a explicar a seguir:

Inicialmente cada jogador retira uma carta do baralho para escolher quem começa a jogar. O jogo começa dividindo igualmente o baralho pelos jogadores presentes.

As cartas que cada jogador recebeu serão a pilha de cartas com que ele vai poder jogar. Estas ficarão viradas para baixo e cada jogador retira da pilha de cartas que lhe foi atribuída as primeiras 5.

Noção base: Uma linha são 2, 3, 4 ou 5 cartas numa linha ou coluna na qual as cartas têm a mesma côr ou cores diferentes e na qual as cartas têm a mesma forma ou formas diferentes. Uma linha com 5 cartas é um Q!nto. O primeiro jogador começa o jogo. A seguir joga o jogador à sua esquerda. Quando na vez de um jogador, este pode fazer uma de três coisas:

1. Adicionar 1, 2, 3, 4 ou 5 peças à grelha numa única linha reta e depois retirar do topo da sua pilha de cartas o número necesário para voltar a ficar com 5 na mão*. Se, numa das jogadas, o jogador completar um ou mais Q!ntos, este deve voltar a completar a sua mão* e deve jogar outra vez.

2. Trocar uma ou mais cartas da sua mão por wildcards já jogadas e/ou a carta Q!nto de forma a que as cartas trocadas continuem a funcionar naquela posição.

3. Passar a vez... e trocar algumas, todas ou nenhuma das susas cartas colocando-as no fundo da sua pilha e retirando do topo o número de cartas necessário para voltar a ficar com 5*.

*ou menos, se não houver cartas suficientes para perfazer 5.

O jogo termina quando houver 1 jogador que fica sem cartas, sendo este o vencedor.

Nota: Uma carta "wild" ou "Q!nto" pode representar cartas diferentes em diferentes direções. No caso de representar coisas diferentes e não ter na mão uma carta que também as consiga representar, então, essa carta é insubstituível.

4 Lógica de Jogo

4.1 Representação do estado de jogo

Embora não haja tabuleiro neste jogo, existe sempre a necessidade de saber a posição relativa entre as cartas que estão em jogo. Como tal, implementamos um tabuleiro na forma de lista de listas de cartas que funciona de forma dinâmica, ou seja, sempre que é introduzida uma nova carta é feita a verificação aos quatro lados do tabuleiro e adicionada uma linha/coluna se assim for necessário, por forma a garantir que existem sempre casas nas bordas do tabuleiro que estão livres para que o jogador as possa utilizar.

O estado do jogo não é apenas representado pelo tabuleiro. A cada jogada é apresentada no ecrã a mão de que o jogador dispõe para jogar. Como sabemos ele apenas tem acesso a cinco das cartas do seu baralho, e durante a sua jogada não lhe são dadas mais cartas para além daquelas cinco.

4.2 Visualização do Tabuleiro

Sendo o tabuleiro uma lista de listas de tamanho variável, os predicados responsáveis pela visualização do tabuleiro de jogo terminam quando a cauda da lista é uma lista vazia nas chamadas recursivas como mostrado abaixo.

Alguns dos predicados utilizados não estão mostrados acima, como por exemplo o predicado responsável pela impressão dos índices das linhas, no entanto estes estão incluídos no código fonte em anexo a este relatório.

4.3 Execução de Jogadas

No início de cada jogada é perguntado ao jogador que tipo de jogada pretende executar. Como sabemos acima o jogador pode optar por adicionar cartas, trocar cartas já presentes no tabuleiro ou trocar cartas que tem na mão.

4.3.1 Adicionar Cartas

Depois de optar por adicionar cartas no tabuleiro, é mostrado no ecrã o estado atual do tabuleiro e as cartas que o jogador tem na mão. O jogador opta pelas cartas que quer jogar bem como a casa do tabuleiro onde pretende que as cartas sejam colocadas. Depois de introduzir esses dados é feita a verificação das condições que têm de ser cumpridas:

- Verifica se a casa está vazia;
- Verifica se tem pelo menos uma carta nas casas adjacentes (excepto no caso de ser a primeira jogada);
- Verifica se, com a carta colocada, a linha fica com menos de seis cartas;
- Verifica também se as regras relacionadas com as cores e naipes se verificam na linha;
- Verifica se, com a carta colocada, a coluna fica com menos de seis cartas;
- Verifica também se as regras relacionadas com as cores e naipes se verificam na coluna.

Acima encontra-se parte do predicado responsável pelas verificações mencionadas que garantem que todas as regras são cumpridas.

4.3.2 Trocar Cartas

Depois de optar por alterar cartas presentes no tabuleiro, tal como no caso acima, é mostrado no ecrã o estado atual do tabuleiro bem como as cartas de que o jogador dispõe. Depois de optar pelas cartas e casas nas quais querem jogar são feitas as seguintes verificações:

- Verifica se a carta escolhida pelo jogador é *Wild*, seja de côr ou naipe;
- Verifica se a casa escolhida pelo utilizador está, de facto, ocupada;
- Verifica se as regras referentes ao número, cores e naipes da sequência são cumpridas na linha;
- Verifica se as regras referentes ao número, cores e naipes da sequência são cumpridas na coluna.

Acima encontra-se parte do predicado responsável pelas verificações mencionadas que garantem que todas as regras são cumpridas.

4.3.3 Trocar cartas

Ao optar por trocar cartas é dada oportunidade ao jogador para escolher quais as cartas (dentro daquelas que estão nesse momento na sua mão) que este pretende trocar. Após a escolha estas cartas passam para o fim do baralho e são repostas na mão por aquelas que estão imediatamente no topo do baralho.

4.4 Avaliação do estado do Jogo

Tendo em conta que nesta modalidade o jogo termina quando um dos jogadores termina o seu baralho, podemos avaliar o estado do jogo através do número de cartas restantes nos baralhos dos jogadores que são impressos nas jogadas.

4.5 Final do Jogo

O jogo termina mal um dos jogadores termine com o seu baralho, tendo sido bem sucedido em deixar todas as suas cartas no tabuleiro sem infringir nenhuma das regras. No final de cada jogada é feita a verificação de se o baralho do jogador está, ou não, vazia. No caso do baralho estar vazio termina o jogo imediatamente.

5 Interface com o Utilizador

6 Conclusões

Este trabalho realizado no âmbito da Unidade Curricular de Programação em Lógica serviu positivamente para que ganhássemos bastantes conhecimentos ao nível da linguagem de PROLOG. Achamos que, tendo um pouco mais de tempo para a conclusão do trabalho, este seria concluído de melhor forma. Mesmo assim o balanço que fazemos é bastante positivo.

Surpreendeu-nos a dificuldade de algumas das regras exigidas por este jogo, bem como a necessidade de fazer um tabuleiro de forma dinâmica, no entanto, encaramos estas dificuldades como desafios que fizemos por superar e fizemo-lo com sucesso. Infelizmente não tivemos tempo para implementar o *bot*, melhoria que proporíamos num futuro em que pudéssemos melhorá-lo.