Universidade de Évora
Curso de Engenharia Informática – 1º Semestre

Programação I

Relatório do Trabalho Prático

2013/2014

A Poção Mágica

Tiago Ávila – nº 31565

João Calhau – nº31621

André Figueira – nº31626

Relatório

Este trabalho a ser feito este ano no âmbito da cadeira de Programação I trata-se de um jogo chamado “A Poção Mágica” este jogo consiste numa vila (uma matriz), em que o utilizador introduz as coordenadas dos habitantes, e das poções e o programa tem de então, encontrar o caminho correto ate essas mesmas. Este trabalho foi divido em partes, essas partes então são:

Main

Este ficheiro, ‘Main.py’, tem como função chamar todas as funções desenvolvidas, no nosso caso guardadas em dois ficheiros distintos: ‘Inputs.py’ e ‘Algorithm.py’. Estes dois ultimos ficheiros serão explicados mais à frente, por agora expliquemos o ficheiro ‘Main’.

Começemos pelo início, a primeira coisa que este programa faz é chamar os dois outros ficheiros atravez de imports:

- import Algorithm
- import Inputs

De seguida, temos um input que vai permitir ao utilizador decidir entre inserir as coordenadas da vila ele próprio, ou se, em vez disso, pretende inserir um ficheiro com tudo. Se o utilizador escolher a opção 1, inserir as coordenadas, o programa passa para o primeiro ‘if’, “if opçao==’1’:” e a partir daí pede ao utilizador a dimensão da matriz, que, por sua vez, vai pedir à função ‘Matriz’, pertencente ao ficheiro ‘Inputs.py’, que crie uma matriz com o tamanho que o utilizador inserir. A variável vila entra depois num ciclo ‘while’ que não deixa a matriz criada ser maior que 7 e menor que 2, e vai pedir ao utilizador o tamanho da matriz até este ter escolhido um número entre esses os dois. Se o valor que o utilizador inseriu está correcto o programa chama então a função ‘habitantes’ e imprime a vila através da função ‘ImprimirVila’, a função ‘habitantes’ irá permitir o input do número de habitantes e depois as coordenadas (será explicado mais detalhadamente mais à frente). Ambas as funções se encontram no ficheiro ‘Inputs.py’.

Na linha de código seguinte é criado um dicionário, dicionário este que vai servir para armazenar os caminhos todos de todos os habitantes. Nas linhas seguintes entramos em 2 ciclos ‘for’ que servem para percorrer todos os pares (linha, coluna) de forma a inseri-los na função ‘Vizinhos’ do ficheiro ‘Algorithm.py’ para encontrar os caminhos. O primeiro ‘if’ dentro dos dois ciclos ‘for’ serve para ver se o par (linha, coluna) é um habitante ou uma coordenada vazia, se for uma coordenada vazia a expressão ‘continue’ faz com que o ciclo passe para a próxima iteração de modo a não usar esse par (linha, coluna) como um habitante. O ‘if’ seguinte verifica se o par (linha, coluna) a ser utilizado já existe, se existir passa para a próxima iteração do ciclo, caso contrário é criada uma chave no dicionário com o valor retomado pela função ‘Vizinhos’, neste caso vai ser uma lista com os caminhos.

As 5 linhas de código seguintes servem apenas para imprimir os caminhos presentes no dicionário, visto que não conseguimos encontrar uma maneira de encontrar o caminho certo de entre todos os caminhos que retomavam da função 'Vizinhos'.

Passando agora para a opção 2, inserir um ficheiro. Se o utilizador escolher a opção 2, ser-lhe-á perguntado qual o nome do ficheiro que pretende abrir. De seguida o programa lê o ficheiro, guarda as coordenadas numa lista e a dimensão da vila e o número de habitantes noutra lista. Depois segue-se a verificação do tamanho e do número de habitantes da matriz, quando o número de habitantes e o tamanho da matriz estiverem bem inseridos o programa faz o mesmo que a função 'habitantes' mas em vez de pedir ao utilizador outra vez em caso de se ter enganado, pede para verificar o ficheiro inserido. De seguida imprime a vila com as especificações inseridas através da função 'ImprimirVila'. Entra em dois ciclos 'for' iguais às linhas de código em cima e chama a mesma função, 'Vizinhos'. E imprime o respectivo dicionário com os caminhos todos. As linhas de código seguintes são uma repetição das anteriores, mas dentro de um ciclo for para se repetir caso o utilizador ache necessário.

Algorithm

Começo já por explicar que estas funções foram baseadas num algoritmo, cujo nome nos é desconhecido, mas cujo autor é Collin J. Simpson. Este algoritmo com que nos baseamos, foi feito numa linguagem mais antiga do python, e tinha uns certos problemas, portanto cabeu a nos adapta lo e "repara lo" de forma a funcionar como nos queriamos.

É definida uma variável e 3 funções neste ficheiro. A variável 'dirs' é uma lista com todas as direcções que as funções abaixo podem usar para verificar se o caminho de um habitante à respectiva poção existe. A primeira função definida é a 'isValid', tudo o que esta função faz é verificar se o par (linha, coluna) actual se encontra dentro da matriz, ou seja, se são um par válido.

Explicarei a terceira função antes da segunda, a função 'Vizinhos2'. Esta função recebe como variáveis a 'vila' (a matriz criada na parte *main* do programa), o nodo (valor da posição do Habitante, ou poção visto que são o mesmo, por exemplo o habitante 1 teria como valor 1, visto que é o primeiro), a 'linha' e a 'coluna' que estão a ser utilizadas de momento, a lista 'Caminho', e uma lista 'CaminhoActual', e as coordenadas iniciais da linha e da coluna (InitL, InitC). Esta função tem como objectivo encontrar todos os caminhos de um habitante. A função começa com um 'if' que vai verificar se o lugar onde se encontra já foi percorrido, caso positivo não retoma nada, caso negativo adiciona à lista 'CaminhoActual' o par (linha, coluna). De seguida encontramos outro 'if' que vai verificar se o lugar onde se encontra é um lugar vazio (estes tem um valor '0') ou se é o lugar onde começou, caso positivo define-se outra variável que vai ser utilizada mais tarde 'lastVal' que tem o valor do lugar actual, depois dá o valor de '#' ao lugar actual para o marcar como já percorrido. De seguia entramos num ciclo 'for' que explorar todas as direcções possíveis a partir da lista definida no inicio 'dirs', dentro deste ciclo verifica-se se a direcção que estamos a explorar é possível ou não, ou seja, se se encontra dentro da matriz e por isso ele invoca a função 'isValid' se for válido entra num ciclo recursivo que vai averiguar essa direcção até sair fora da matriz, depois passa à próxima direcção e verifica-a até sair fora da matriz e assim sucessivamente até ter percorrido todas as

direcções quando o ciclo 'for' tiver terminado, é retomado o valor anterior ao lugar actual sendo este o que foi guardado anteriormente na variável 'lastVal'. No 'elif' seguinte a função vai verificar se o valor do lugar actual é a poção ou não.

A segunda função serve apenas para criar uma lista 'Caminho' vazia cada vez que esta é chamada e para chamar a função 'Vizinhos2' com os valores da 'vila', 'nodo', 'linha', 'coluna' que foram previamente inseridos nesta e com a lista vazia 'Caminho' definida anteriormente. Serve ainda para definir a lista 'CaminhoActual' da função 'Vizinhos2' como uma lista vazia '[]' e para definir as coordenadas iniciais de initL e initC como as coordenadas inseridas nesta função. No fim retoma a lista 'Caminho' para ser depois utilizada no 'Main.py'

Inputs

Os inputs começam com a função "habitantes" onde a função pede o input, este input como é para definir o número de habitantes antes de qualquer input passar a fase seguinte da função tem que ser garantido que se tratar de um inteiro assim é chamada a função "e_digito", verifica se o input é um algarismo de 0 a 9.

Quando garantido que o algarismo é um inteiro a função habitantes entra num ciclo de while onde é verificado se o número de habitantes é maior que o número que o número de colunas, caso seja volta a ser pedido um valor para os habitantes onde é de novo verificado se é dígito pela função "e_digito" isto ocorre até que o número de habitantes seja igual ou inferior ao número de colunas da matriz.

Após a saída do ciclo while entra num ciclo for a função "habitantes",

A função "matriz" será onde o tamanho da vila será construído, esta função funciona da seguinte maneira: É criada uma lista com o nome de "colunas" e outra com o nome "linhas"

Ocorre um ciclo "for" onde percorre o input inteiro feito anteriormente para a vila de 0 até o inteiro de input menos 1, e outro ciclo for que faz o mesmo. o segundo ciclo de for compõem as linhas de 0 até n-1 espaços onde e adicionada o "0" formando tendo assim um lista chamada" linhas" com tantos "0" quanto o comprimento de linha da matriz, já fora do segundo for, o primeiro for percorre o input vila de o ate n-1 onde vai adicionando a lista com o nome "colunas" n vezes o contido da lista "linhas" formando assim uma matriz quadrada.

A função "get_linha" e "get_linha" combinadas encontraram um ponto na matriz.

A função "get_linha" encontra exactamente a linha dada pela coordenada "x", podemos então procurar a coluna (coordenada "y") esta segunda coordenada é dada pela função "get_coluna", esta função recebe as variáveis x, y e vila criar uma variável "linhas" onde chama a função "get_linha" para saber a coordenada "x" na matriz após isso procura na linha da coordenada "x" o ponto da coordenada "y".

Estas duas funções são utilizadas no projecto para conseguir saber se as casa onde se coloca o habitante ou a poção se estão ocupadas.

Após serem introduzidas as coordenadas, primeiramente tem que ser testadas para saber se estão de acordo. Este processo foi feito anteriormente para o tamanho da vila a para o tamanho de habitantes pela função “e_digito” esta função já falada anteriormente não pode ser utilizada neste casa porque o input das coordenadas tem que ser uma sequência de dígitos ia reconhece-los como não dígitos por o valor de input ser maior que 9.

Assim e necessário criar outra função para que as coordenadas sejam testadas esta função da pelo nome de “int_coord” esta função percorre a uma lista onde estão introduzidas as coordenadas e com um ciclo “for” testa as 4 primeiras coordenadas, ou seja, os 4 primeiros valores da lista se estes forem digito aceita o input se não volta a pedir as coordenadas até que estejam de acordo (4 dígitos seguidos).

Após a verificação das coordenadas, estas são introduzidas na função “insert”, esta função vê-se a casa já esta ocupada ou não, da seguinte maneira. A “ elem” criada na função “insert” chamada a função “get_coluna” onde são introduzidas as coordenadas respectivas aos inputs testado anteriormente pela função “int_coord”, se o local correspondente as coordenadas for diferente de 0 ele pede novas coordenadas e testa as com a função “int_coord” volta a criar a variável “elem” agora com as novas coordenadas e volta a ver se o local esta diferente de “0” após conseguir encontrar esse local diferente de zero e chamada a função “posições_vila” que tem como função inserir na matriz na posição vazia o numero correspondente a poção ou ao habitante. Esta função “posições_vila” porta-se da seguinte maneira recebe as coordenadas anteriormente testadas e vai encontrar na vila o local das coordenadas e atribuir um valor inteiro correspondente ao habitante ou a poção.

Função “ImprimirVila” através de um ciclo de “for” imprime linha a linha a matriz já com os habitantes e as poções colocados.

Conclusão

Como já dito anteriormente, este código não está completo uma vez que lhe falta a ultima parte do trabalho (intersectar os caminhos e imprimir a matriz com os caminhos respectivos, se possível)

Em consequência decidimos então, fazer com que o programa imprima a matriz com a posição dos habitantes e poções, e todos os caminhos que os habitantes podem percorrer.