Universidade Autónoma de Lisboa

Licenciatura Engenharia Informática

Algoritmia e Programação

Projeto Batalha Naval

Autores:Duarte Elvas 30005521 João Gil 30005106 Tiago Leitão 30005272

Grupo: 01 Turma B

2019/2020

Índice

[1](#_heading=h.3znysh7) Introdução 3

[2](#_heading=h.2et92p0) Metodologia 3

[3](#_heading=h.3dy6vkm) avaliação 5

[4](#_heading=h.1t3h5sf) Conclusões 5

[5](#_heading=h.4d34og8) Bibliografia 5

Versões

|  |  |  |
| --- | --- | --- |
| Versão | Data | Descrição |
| 1 | DD/MM/YYYY | Descrição sumária. |

# Introdução

O nosso objetivo com este trabalho é criar um jogo de batalha naval em Python, no qual dois jogadores poderão realizar vários jogos um contra o outro e no final serão anotadas todos os jogos ocorridos e as vitórias de todos os jogadores.

# Metodologia

Neste tópico vamos descrever o nosso raciocínio perante as funções que achamos mais complexas, porque as outras acabam por ser simples.

A base do nosso trabalho, está contida no Dicionário Geral, ou “DG”:

*def* Dicionario\_Geral():

Neste dicionário está contida toda a informação inerente ao nosso projeto.

Ele é a nossa base de dados, todas as funções estão, de alguma maneira, ligadas a este dicionário.

A primeira função registrar jogador:

*def* Registar\_Jogadores(*DG*,*nome*):

DG["jogadores"][nome]={

"vitorias":0,

"Jogos":0,

}

return DG

A função Registar Jogadores adiciona mais um dicionário ao dicionário geral, inserindo o input do utilizador como key no dicionário. Este novo dicionário contém as vitórias e os jogos jogados pelo jogador.

A função iniciar jogo é uma das mais funções complexas do nosso projeto, sendo que é ela que adiciona não só os tabuleiros a cada jogador, mas também todos os registos que cada um vai utilizar no futuro.

A função Translator:

*def* translator(*letra*):

return ord(letra)-65

Esta função traduz a letra inserida pelo jogador para definir a coluna do tabuleiro para um número entre 0 a 9( por exemplo A=0,B=1,C=2,etc..)

*def* bloco(*DG*,*nome*,*linha*,*coluna*): #verifica se o jogador pode ou não coloc a=[[0,1],[0,-1],[1,0],[-1,0],[-1,-1],[1,-1],[1,1],[-1,1]]

permite=True

result=[]

tabuleiro=DG["jogadores\_em\_jogo"][nome]["tabuleiro"]

posiçao=[linha,coluna]

for i in a:

if not posiçao[0]+i[0]<0 and not posiçao[0]+i[0]>9 and not \ posiçao[1]+i[1]<0 and not posiçao[1]+i[1]>9:

if tabuleiro[posiçao[0]+i[0]] [posiçao[1]+i[1]]!=0:

permite=False

result.append([posiçao[0]+i[0],posiçao[1]+i[1]])

result.insert(0,permite)

return result

A função bloco é sem dúvida a jóia da coroa do nosso projeto,(sabemos que o nome não se adequa mas decidimos mantê-lo pois facilitava a nossa leitura de algumas funções) esta função é chamada em duas funções distintas: na “vericar\_posição\_em\_tabuleiro” e na “Remover\_Navios”.

A Bloco foi programada de modo a ter dois outputs diferentes, para se ajustar às duas funções em que está inserida: O Output da Bloco vai ser uma lista, em que a primeira posição é um boolean que descreve se a posição que lhe foi dada em argumento têm ou não um navio nas suas redondezas.Se tiver, a função bloco faz o return False e das posições em que o navio se encontra. Pelo contrário, se não encontrar nenhum navio, faz apenas o return True.

A Função vericar\_posição\_em\_tabuleiro verifica todas as posições em que o navio vai ser colocado usando o bloco e verificando se a posição está dentro das bordas do tabuleiro.

A Função colocar navios, coloca os navios no tabuleiro com um id unico, esta função chama a função “vericar\_posição\_em\_tabuleiro”.

A Função remover\_navios, remove os navios por recursão, aproveitando a função bloco, que retorna as posições dos barcos adjacentes á posição indicada. Assim em quando o primeiro return do bloco for False, a função remover\_navios vai continuar a correr e a substituir todas as posições indicadas pelo bloco por 0 (água).

A Função Removido é complementar à função remover\_navios, sendo que esta adiciona 1 à quantidade dos navios que removeu.

A Função tiro é uma das menos complexas, sendo que esta apenas verifica a posição inserida pelo utilizador e adiciona ou remove quantidades na biblioteca.

A função Desistir acaba o jogo no caso que um jogador decida desistir. Quando esse jogador desiste, o jogo acaba e é adicionado ao dicionário do jogador mais um jogo jogado. Por sua vez, ao seu adversário é adicionado mais um jogo e uma vitória. Também é possivel os dois jogadores desistirem. Neste caso é apenas adicionado mais um jogo ao dicionário de cada jogador.

Por fim gostaríamos de explicar melhor a composição do dicionário:

"jogadores":{

# "jogador1":{

# "vitorias":0,

# "Jogos":0,

# }

},

Neste “Sub-Dicionário” do dicionário geral são armazenados os jogadores e respetivas vitórias e derrotas.

"jogo\_em\_curso":False,

"combate\_em\_curso":False,

"Ronda":"",

Aqui são armazenados os estados de jogo, se o jogo está em curso, se o combate está em curso, e o nome do jogador que jogou pela última vez.

"jogadores\_em\_jogo":{

#"jogador1":{

#"tabuleiro":[],

#"Frota":{}

# }

},

O jogadores\_em\_jogo armazena dicinários que têm como key o nome do respectivo jogador, aqui são armazenados os tabuleiros e a frota dos jogadores.

DG["jogadores\_em\_jogo"][jogador]["Tiros"]=0

DG["jogadores\_em\_jogo"][jogador]["Tiros\_em\_navios"]=[]

DG["jogadores\_em\_jogo"][jogador]["Navios\_afundados"]=0

Aqui também estão armazenados os dados relativos ao jogo, tais dados que serão posteriormente mostrados com o comando “V”. O Tiros\_em\_navios é uma lista porque é aqui onde vão ser armazenadas as posições dos navios que já foram acertados.

DG["jogadores\_em\_jogo"][jogador]={"tabuleiro":[[0 for \_ in range(0,10)] for \_ in range(0,10)]}

O Tabuleiro vai ser criado posteriormente no “Iniciar Jogo”.

DG["jogadores\_em\_jogo"][jogador]["Frota"]={

"L":{

"quantidade":4,

"navios\_em\_jogo":4,

"contador":0,

"Frota\_em\_jogo":{

#"L1":1,

#"L2":1,

#"L3":1,

#"L4":1 }

},

A frota, tal como o tabuleiro vai ser criada no inicar jogo.

Aqui podemos observar que o dicionário “Frota” se vai ramificar em mais alguns dicionários, um para cada tipo de navio encontrado no jogo.

Cada dicionário ramifica-se em mais 4:

A quantidade indica a quantidade de navios ainda disponíveis para colocar no tabuleiro, sendo que só se pode iniciar o combate quando todas as quantidades se encontram a 0.

Os navios em jogo são os navios que ainda se encontram vivos no tabuleiro, sendo que o jogo acaba se o valor de navios\_em\_jogo de todos os tipos de navio de um certo jogador forem 0.

O contador vai aumentar 1 sempre que um navio novo é colocado. Assim, garante que cada navio colocado no tabuleiro tem um id diferente.Isto é necessário para conseguirmos verificar a quantidade de navios ainda vivos de cada tipo.

A Frota em jogo contem como keys o id de cada navio do tipo indicado, e a quantidade de “Letras” que o mesmo ainda têm vivas. Quando qualquer um destes valores chega a 0, é subtraído um aos navios\_em\_jogo.

# Avaliação

Conseguimos realizar todas as diferentes funcionalidades do programa no qual consiste o nosso projeto com sucesso, tendo que em todos os testes que realizámos ocorreram os resultados desejados (ou seja ocorreram todas as saídas esperadas).

Achamos que conseguimos realizar tão rápido e tão eficientemente o projeto devido ao nosso método de organização, pois antes de começarmos a programar já tínhamos todas as funções definidas em papel. É por isso que bastantes funções são reutilizadas e que apenas usamos dois tabuleiros na realização do trabalho.

# Conclusões

Conseguimos realizar o projeto com sucesso, conseguindo desta forma realizar todos os testes pedidos com sucesso.

Com este projeto adquirimos um novo senso de trabalho em equipa, pois este foi um projeto longo e trabalhoso e sem uma boa gestão de trabalho entre nós teria sido muito mais difícil de o realizar com sucesso.

Também adquirimos novas capacidades em termos de programação de Python, pois foram necessárias várias novas ferramentas para conseguir criar todas as funções necessárias para a realização deste projeto. O projeto foi realizado com recurso ao Visual Studio Code, logo também ganhamos um melhor entendimento deste editor de código.

Com base em todo o trabalho realizado, podemos afirmar que estamos satisfeitos com o resultado do nosso trabalho e esperemos que o professor também fique.