# Instituto Politécnico do Cávado e Ave



# Trabalho Prático 1

# Relatório

# Jorge Miguel Arezes Noro | 15705

# Engenharia e Desenvolvimento de Jogos Digitais

# Paradigmas de Programação II

# Docente: Luís Gonzaga Martins Ferreira

2º Semestre

1. Estruturas de Dados Estáticas

Para o desenvolvimento da parte 1 do trabalho prático, a estrutura de dados estática capaz de armazenar todos os dados do cartão matriz será um *array* de inteiros. Uma possível solução, a que foi implementada neste trabalho, passou por separar as células do cartão matriz em *structs* chamados de *Nodes.* Cada *Node* terá um *array* de inteiros, em que será armazenada a sequência de números.

Para construir a matriz, usamos então outro *struct* chamado de *Row.* Este *struct* armazena o número de *Nodes* igual ao número de colunas definidas na matriz (ver *Types.h)*. O *struct* *Matrix* tem como membro um *array* do tipo *Row*, em que armazena o número de linhas definidas na matriz. É possível ver na imagem abaixo a relação dos *structs* entre si:

|  |
| --- |
|  |
| Figura 1-Relação das estruturas de dados. |

A solução pode parecer complexa demais para a resolução deste problema simples, mas desta forma deixamos espaço para o crescimento da aplicação, como por exemplo, adicionar informação sobre determinado *Node* ou sobre determinada *Row*. Uma solução mais simples e básica podia ser implementar simplesmente um *array* tridimensional:

|  |
| --- |
|  |
| Code Snippet 1- Possível implementação da matriz, usando um simples array tridimensional |

As funções de alocação de memória destes objetos são encontradas no ficheiro *ObjectCreation.c*. Fica abaixo um exemplo (alocação em memória de uma matriz):

|  |
| --- |
|  |
| Code Snippet 2- Alocação em memória de uma matriz. |

O código chama a função *createRow()*, que por sua vez chama a função *createNode()*. Esta, vai popular o *Node* com uma sequência aleatória de números. Podemos ver essa função abaixo:

|  |
| --- |
|  |
| Code Snippet 3- Alocação em memória de um Node |

No que diz respeito a algoritmos, esta primeira parte do trabalho é relativamente simples. Podemos analisar como exemplo, a função em que pedimos ao utilizar que valide uma operação. Fica abaixo o código em excerto (algumas partes foram retiradas para auxiliar a leitura neste relatório, ver ficheiro *UtilityFunctions.c* para analisar o código completo):

|  |
| --- |
|  |
| Code Snippet 4- Função para validar uma operação |

A função gera números aleatórios para a linha, coluna e posição do dígito pretendido. De seguida, mostra ao usuário qual a célula e posição do dígito que se pretende e obtém a resposta. É chamada uma função que valida essa resposta. Vejamos abaixo:

|  |
| --- |
|  |
| Code Snippet 5- Função para validar escolha do utilizador |

Ao realizar este trabalho, houve como ideal apelar à parametrização do programa. Existem várias variáveis que podem ser ajustadas conforme as preferências do utilizador. Ficam abaixo ditas variáveis com a descrição do seu papel no programa:

|  |
| --- |
|  |
| Code Snippet 6- Variáveis para parametrização do programa |

2. Estruturas de Dados Dinâmicas