Ficha 6

Programação Imperativa

Buffers

1. Considere o seguinte tipo para representar *queues* de números inteiros.

```
#define MAX 100
typedef struct queue {
    int inicio, tamanho;
    int valores [MAX];
} QUEUE;
```

   Defina as seguintes funções sobre este tipo:

   (a) `void initQueue (QUEUE *q)` que inicializa uma queue (passa a representar uma queue vazia)

   (b) `int isEmptyQ (QUEUE *q)` que testa se uma queue é vazia

   (c) `int enqueue (QUEUE *q, int x)` que acrescenta `x` ao fim de `q`; a função deve retornar `0` se a operação fôr feita com sucesso (i.e., se a queue ainda não estiver cheia) e `1` se a operação não fôr possível (i.e., se a queue estiver cheia).

   (d) `int dequeue (QUEUE *q, int *x)` que remove de uma queue o elemento que está no início. A função deverá colocar no endereço `x` o elemento removido. A função deverá retornar `0` se a operação for possível (i.e. a queue não está vazia) e `1` em caso de erro (queue vazia).

   (e) `int front (QUEUE *q, int *x)` que coloca no endereço `x` o elemento que está no início da queue (sem modificar a queue). A função deverá retornar `0` se a operação for possível (i.e. a queue não está vazia) e `1` em caso de erro (queue vazia).

2. Na representação de queues sugerida na alínea anterior o array de valores tem um tamanho fixo (definido pela constante `MAX`). Uma consequência dessa definição é o facto de a função de inserção (`enqueue`) poder não ser executada por se ter excedido a capacidade da estruturas.

   Uma definição alternativa consiste em não ter um array com tamanho fixo e sempre que seja preciso mais espaço, realocar o array para um de tamanho superior (normalmente duplica-se o tamanho do array).

   Considere então a seguinte definição alternativa e adapte as funções definidas atrás para esta nova representação.

   Use as funções `malloc` e `free` cujo tipo está definido em `stdlib.h`.

```
typedef struct queue {
    int size; // guarda o tamanho do array valores
    int inicio, tamanho;
    int *valores;
} QUEUE;
```