exercicio-tad-vetor-parte1.md 9/12/2019

Tipo Abstrato de Dados (Vetor)

Desenvolver um tipo abstrato de dados para armazenar e manipular vetores de valores inteiros.

**DADOS**

Os dados do vetor serão organizados em uma *struct* denominada Vetor.

typedef struct{ int\* vetor; int size; //quantidade de elementos armazenados no vetor int length; // tamanho alocado } Vetor;

Vamos utilizar outras duas definições para nos auxiliar na implementação e manutenção do código.

O tipo DataType será definido representar o tipo de dado manipulado pelo TAD Vetor. Neste exemplo, o tipo manipulado é o int. Caso queiramos utilizar outro tipo, basta alterar essa linha.

typedef int DataType;

O segundo tipo definido é o Boolean. Como a linguagem C não possui um tipo primitivo para representar valores booleanos, criamos um tipo enumerado para permitir o uso dos valores true e false. Dessa forma, quando precisarmos criar uma função booleana, basta definir o retorno com o tipo Boolean e devolver os valores true ou false.

typedef enum {false, true} Boolean;

Crie uma constante que será usada como tamanho padrão para a alocação do vetor dentro da struct.

#define LENGTH 5

**OPERAÇÕES**

Todas as operações previstas para o vetor serão mapeadas para funções. Todas as funções serão desenvolvidas especificamente para manipular a estrutura definida Vetor.

**Importante:** *Você deve ser capaz de validar se sua função funciona ou não. Para isso, crie pequenos casos de teste para validar as funções desenvolvidas.*

1 / 4

exercicio-tad-vetor-parte1.md 9/12/2019

Para facilitar a identificação, utilizaremos o prefixo "vetor\_" para cada função.

**PARTE 1**

**Criar o vetor**

Vetor\* vetor\_new();

**Destruir o vetor**

void vetor\_free(Vetor\* v);

**Imprimir o vetor**

void vetor\_print(Vetor\* m);

**Adicionar um elemento em uma posição do vetor**

A funcionalidade de inserção deve providenciar espaço sempre que necessário. Crie uma função auxiliar para isso e utilize a estratégia de dobrar o tamanho do vetor sempre que houver necessidade de espaço.

Boolean vetor\_insert(Vetor\* v, DataType element, int index);

**Adicionar um elemento no final do vetor**

A funcionalidade de inserção deve providenciar espaço sempre que necessário. Crie uma função auxiliar para isso e utilize a estratégia de dobrar o tamanho do vetor sempre que houver necessidade de espaço.

Boolean vetor\_add(Vetor\* v, DataType element);

**Remover um elemento do vetor de uma posição específica**

A remoção deve se preocupar com a quantidade de espaço ocioso no vetor. Caso a remoção resulte em uma taxa de ocupação abaixo de 25%, o tamanho do vetor deve ser reduzido pela metade. Crie uma função auxiliar para verificar e efeturar a redução.

2 / 4

exercicio-tad-vetor-parte1.md 9/12/2019

DataType vetor\_remove1(Vetor\* v, int index);

Boolean vetor\_remove2(Vetor\* v, int index, DataType\* ptr);

**Remover o elemento da primeira posição do vetor**

A remoção deve se preocupar com a quantidade de espaço ocioso no vetor. Caso a remoção resulte em uma taxa de ocupação abaixo de 25%, o tamanho do vetor deve ser reduzido pela metade. Crie uma função auxiliar para verificar e efeturar a redução.

DataType vetor\_shift1(Vetor\* v);

Boolean vetor\_shift2(Vetor\* v, DataType\* ptr);

**Acessar um elemento do vetor**

Essa funcionalidade deve ser disponibilizada por meio de três funções. As funções provêm o mesmo comportamento, porém, de formas distintas.

DataType vetor\_get1(Vetor\* v, int index);

Boolean vetor\_get2(Vetor\* v, int index, DataType \*ptr);

DataType\* vetor\_get3(Vetor\* v, int index);

**Alterar um elemento no vetor**

Boolean vetor\_set(Vetor\* v, int index, DataType valor);

**Aplicar uma alteração à todos os elementos do vetor.**

Essa funcionalidade pode ser obtida por meio de uma função que recebe por parâmetro um ponteiro da função que realizará a alteração em cada elemento. Essa funcionalidade já foi discutida em sala de aula.

void vetor\_map(Vetor\* v, void (\*funcao)(DataType\*));

A função recebe o vetor e o ponteiro da função que será invocada para cada elemento.

3 / 4

exercicio-tad-vetor-parte1.md 9/12/2019

**Criar um sub‒vetor.**

O sub‒vetor pode ser adquirido com base nos índices do vetor.

A primeira função recebe o vetor v e o índice index e cria o sub‒vetor a partir de index. Quando o índice é negativo, ele representará as posições do fim para o inicio. Por exemplo, o índice ́‒1́ representa a última posição do vetor, ‒2 a penúltima e assim por diante.

Vetor\* vetor\_sub1(Vetor\* v, int index);

A segunda função recebe o intervalo desejado para a criação do sub‒vetor.

Vetor\* vetor\_sub2(Vetor\* v, int start, int end);

**Acessar o tamanho do vetor**

int vetor\_size(Vetor\* v);

4 / 4