Estrutura de Dados - MC202 A

1º Semestre de 2018

Tiago de Paula Alves - R.A.: 187679

3 de abril de 2018

Lista 1 - Exercício 2

a)

```c
/* Inicialização dos valores, zerando-os. */
void  vetor_inicializar(int **vec , int  tamanho) {
    size_t i;
    /* percorre o vetor de ponteiros */
    for (i = 0; i < tamanho; i++) {
        /* e inicializa o valor para onde está sendo apontado */
        *(vec[i]) = 0;
    }
}

/* Libera a memória. */
void  vetor_desalocar(int **vec , int  tamanho) {
    size_t i;
    /* percorre o vetor */
    for (i = 0; i < tamanho; i++) {
        /*  liberando a memória de cada ponteiro */
        free(vec[i]);
    }
    /* e então libera a do vetor */
    free(vec);
}
```

b)

Pela assinatura da função `vetor_dobrar_tamanho`, é provável que a sua implementação nem sempre funcione como esperado. Isso é devido a variável `vec`, que recebe seu valor de entrada durante a chamada em uma posição de memória diferente a da função superior, assim, qualquer alteração nela é perdida ao encerrar a chamada, de onde vem a parte inesperada na função. É possível, no entanto, que função funcione se, por exemplo, foi implementada usando `realloc` e, por acaso, o sistema reconheceu que dava para alocar mais memória em sequência com a original e não precisou alterar o valor númerico do ponteiro. Porém, esse problema como um todo pode ser facilmente evitado usando uma referência para o ponteiro do vetor, em vez do ponteiro em si, e ir atualizando o seu valor quando necessário. Outra opção também é retornar o novo ponteiro e esperar que o usuário cuide da atualização do apontador. Existe ainda uma solução mais críptica, possível até de ser implementada com essa assinatura, que é usando variáveis globais, porém não é recomendado, exatamente por dificultar a leitura e ser mais suscetível a modificações inesperadas e, possivelmente, erros.