

# Estruturas de Dados

## Tipos Abstratos de Dados – Exercícios



# Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel,  
*Introdução a Estruturas de Dados*, Editora Campus  
(2004)

Exercícios Parte II (pág. 214)

# Questão 1

- Implemente um tipo abstrato de dados para vetores no R3, contendo:
  - um arquivo com a definição do tipo e das assinaturas das funções
    - cria (cria um vetor)
    - libera (libera o espaço reservado para um vetor)
    - soma (soma de 2 vetores)
    - prodint (produto interno)
    - prodvet (produto vetorial)
  - um arquivo com a implementação do tipo e das funções

# Questão 1

```
typedef struct vetor Vetor;  
  
Vetor* cria(float x, float y, float z);  
  
void libera(Vetor* v);  
  
Vetor* soma(Vetor* v1, Vetor* v2);  
  
float prodint (Vetor* v1, Vetor* v2);  
  
Vetor* prodvet (Vetor* v1, Vetor* v2);
```

# Questão 1

```
#include <stdlib.h>
#include <stdio.h>

#include "vetor.h"

struct vetor {
    float x;
    float y;
    float z;
};
```

# Questão 1

```
Vetor* cria(float x0, float y0, float z0)
{
    Vetor* v=(Vetor*) malloc(sizeof(Vetor));

    v->x=x0;
    v->y=y0;
    v->z=z0;

    return v;
}

void libera(Vetor* v)
{
    free(v);
}
```

# Questão 1

```
Vetor* soma(Vetor* v1, Vetor* v2)
{
    Vetor *v=cria(v1->x+v2->x, v1->y+v2->y, v1->z+v2->z);
    return v;
}

float prodint (Vetor* v1, Vetor* v2)
{
    float prod = v1->x*v2->x + v1->y*v2->y + v1->z*v2->z;
    return prod;
}

Vetor* prodvet (Vetor* v1, Vetor* v2)
{
    Vetor* pv=(Vetor*) malloc(sizeof(Vetor));
    pv->x = v1->y*v2->z - v1->z*v2->y;
    pv->y = -(v1->x*v2->z - v1->z*v2->x);
    pv->z = v1->x*v2->y - v1->y*v2->x;
    return pv;
}
```

## Questão 2

Considere um cadastro de produtos de um estoque, com as seguintes informações para cada produto:

- Código de identificação do produto: representado por um valor inteiro
- Nome do produto: com até 50 caracteres
- Quantidade disponível no estoque: representado por um número inteiro
- Preço de venda: representado por um valor real

(a) Defina uma estrutura em C, denominada produto, que tenha Os campos apropriados para guardar as informações de um produto, conforme descrito acima.

(b) Escreva uma função que receba os dados de um produto (código, nome, quantidade e preço) e retorne o endereço de um struct produto criado dinamicamente e inicializado com os valores recebidos como parâmetros pela função. Essa função deve ter o seguinte protótipo:

```
struct produto* cria (int cod, char* nome, int quant, float preco);
```

## Questão 2

```
struct produto {  
    int cod;  
    char nome[51];  
    int quant;  
    float preco;}  
typedef produto Produto
```

```
Produto* cria (int cod, char* nome, int quant, float preco)  
{Produto *p  
    p = (Produto*)malloc(sizeof(Produto));  
    p->cod = cod;  
    strcpy(p->nome, nome); /* strcpy(destino,origem) */  
    p->quant = quant;  
    p->preco = preco;  
    return p  
}
```

# Questão 3

(Semelhante aos exercícios 6.3 e 6.4)

Considerando as declarações abaixo para representar o cadastro de grupo de alunos de diferentes turmas, implemente funções para criar e imprimir o cadastro.

```
struct aluno { char nome[81];  
               char matricula[8];  
               char turma;  
               float p1;  
               float p2;  
               float p3; };  
typedef struct aluno Aluno;
```

Implemente uma função que tenha como valor de retorno a média final obtida pelos os alunos de uma determinada turma. A nota final de cada aluno é dada pela média das três provas.

```
float media_turma (int n, Aluno** turmas, char turma);
```

## Questão 3

```
/* Rotinas para tratamento da turmas de alunos */
#include <stdio.h>
struct aluno {
    char nome[81];
    char matricula[8];
    char turma;
    float p1;
    float p2;
    float p3;
};
typedef struct aluno Aluno;

#define MAX 100
Aluno* turmas[MAX];
```

## Questão 3

```
void inicializa (int n, Aluno** turmas)
{
    int i;
    for (i=0; i<n; i++)
        turmas[i] = NULL;
}
```

```

void preenche (int n, Aluno** turmas, int i)
{
    if (i<0 || i>=n) {
        printf("Indice fora do limite do vetor\n");
        exit(1);  /* aborta o programa */
    }
    if (turmas[i]==NULL)
        turmas[i] = (Aluno*)malloc(sizeof(Aluno));
    printf("Entre com o nome:");
    scanf(" %80[^\n]", turmas[i]->nome);
    printf("Entre com a matricula:");
    scanf(" %7[^\n]", turmas[i]->matricula);
    printf("Entre com a turma:");
    scanf(" %c", &turmas[i]->turma);
    printf("Entre com a nota de P1:");
    scanf("%f", &turmas[i]->p1 );
    printf("Entre com a nota de P2:");
    scanf("%f", &turmas[i]->p2 );
    printf("Entre com a nota de P3:");
    scanf("%f", &turmas[i]->p3 );
}

```

## Questão 3

```
void imprime (int n, Aluno** turmas, int i)
{
    if (i<0 || i>=n) {
        printf("Indice fora do limite do vetor\n");
        exit(1); /* aborta o programa */
    }

    if (turmas[i] != NULL)
    {
        printf("%s, " ", %c, " ", %f, %f, %f ",
        turmas[i]->matricula,
        turmas[i]->turma,
        turmas[i]->p1, turmas[i]->p2, turmas[i]->p3 );
    }
}
```

## Questão 3

```
void imprime_tudo (int n, Aluno** turmas)
{
    int i;
    for (i=0; i<n; i++)
        imprime(n,turmas,i);
}
```

## Questão 3

```
float media_turma (int n, Aluno** turmas, char turma)
{ int i; float k, soma=0;
  for (i=0; i<n; i++)
    if (turmas[i] != NULL && turmas[i]->turma == turma)
      { soma = soma + (turmas[i]->p1 + turmas[i]->p2 + turmas[i]->p3)/3;
        k = k + 1;
      }
  return soma/k;
}
```

## Questão 3

```
int main (void)
{ float media;
  Aluno* turmas[4];
  inicializa(4,turmas);
  preenche(4,turmas,0);
  preenche(4,turmas,1);
  preenche(4,turmas,2);
  preenche(4,turmas,3);
  media = media_turma(4, turmas, 'a');
  printf("%f",media);
  return 0;
}
```