UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
Bacharelado em Ciência da Computação e
Engenharia da Computação

INF 01203 – Estruturas de Dados

Profa. Renata de Matos Galante (galante@inf.ufrgs.br)

Aula 02 – Tipos Abstratos de Dados

1 – Considere o ***TAD ponto*** apresentado em aula (descrito abaixo):

a) acrescente novas operações ao ***TAD ponto***, como soma e subtração;

b) especifique um programa de aplicação em C que utilize as duas novas funções.

ponto.h

```
/* TAD: Ponto (x,y) */
/* Tipo exportado */
typedef struct ponto Ponto;

/* Funções Exportadas */

/* Cria coordenada */
Ponto* pto_cria (float x, float y);

/* Libera ponto */
void pto_libera(Ponto* p);

/* Acessa ponto */
void pto_acessa(Ponto* p, float* x, float* y);

/* Atribui */
void pto_atribui(Ponto* p, float* x, float* y);

/* Distrância */
float pto_distancia(Ponto* p1, Ponto* p2);
```

ponto.c

```
# include <studio.h>
# include “ponto.h”

struct ponto {
  float x;
  float y;
};

Ponto* pto_cria (float x, float y)
{
  Ponto* p = (Ponto*) malloc(sizeof(Ponto));
  if (p == NULL) {
    printf(“Memória Insuficiente!”\n);
    exit(1);
}
p->x = x;
p->y = y;
return p;
}
```

aplicacao.C

```
# include <studio.h>
# include “ponto.h”

int main (void)
{
  Ponto* p = pto_cria(2.0, 1.0);
  Ponto* q = pto_cria(3.4, 2.1);
  float d = pto_distancia(p,q);
  printf(“Distância entre ponto: %\n”, d);
  pto_libera(q);
  pto_libera(p);
  return 0;
}
```

2 – Considere o ***TAD matriz*** apresentado abaixo:

a) use apenas as operações definidas pelo ***TAD matriz*** e implemente uma função que, dada uma matriz, crie dinamicamente a matriz transposta correspondente. (Em matemática, uma matriz transposta é o resultado da troca de linhas por colunas em uma determinada matriz).

matriz.h

```
/* TAD: matriz m por n */

typedef struct matriz Matriz;

/* Cria matrtiz de dimensão m por n */
Matriz* mat_cria (int m, int n);

/* Libera memória alocada para matriz */
void mat_libera(Matriz* mat);

/* Acessa elemento da linha i coluna j da matriz */
float mat_acessa(Matriz* mat, int i, int j);

/* Atribui o elemento da linha i e da coluna j*/
void mat_atribui(Matriz* mat, int i, int j, float v);

/* Retorna o número de linhas da matriz */
int mat_linhas(Matriz* mat);

/* Retorna o número de colunas da matriz */
int mat_colunas(Matriz* mat);
```

matriz.c

```
# include <studio.h>
# include "matriz.h"

struct matriz {
   int lin;
   int col;
   float* v;
};
```

<<**implementação de todas as funções definidas no **matriz.h >>

3 – Especifique a interface de um TAD para as situações apresentadas a seguir:

a) (Exercício 2.4 – Livro Estruturas de Dados) Considere uma empresa que precisa armazenar os seguintes dados de um cliente: *nome completo*; *ano de nascimento*; *renda mensal do cliente*. Especifique um TAD (***TAD Cliente***) para armazenar os dados de um cliente e as operações necessárias para inserir, consultar e excluir os dados dos clientes (para as operações apenas especifique a interface – cabeçalho das operações). Em seguida, implemente uma função para exibir o número de clientes com renda mensal acima da média, e exibir o número de clientes que nasceram entre 1980 e 2000.

b) (Exercício 2.5 – Livro Estruturas de Dados) Considere um conjunto de informações relativas a alunos, *constituído de nome*, *número de matrícula* e *data de nascimento*. Especifique um TAD (***TAD Aluno***) para armazenar os dados dos alunos e as operações necessárias para inserir, consultar e excluir esses dados. Implemente uma aplicação que utilize o tipo Aluno.